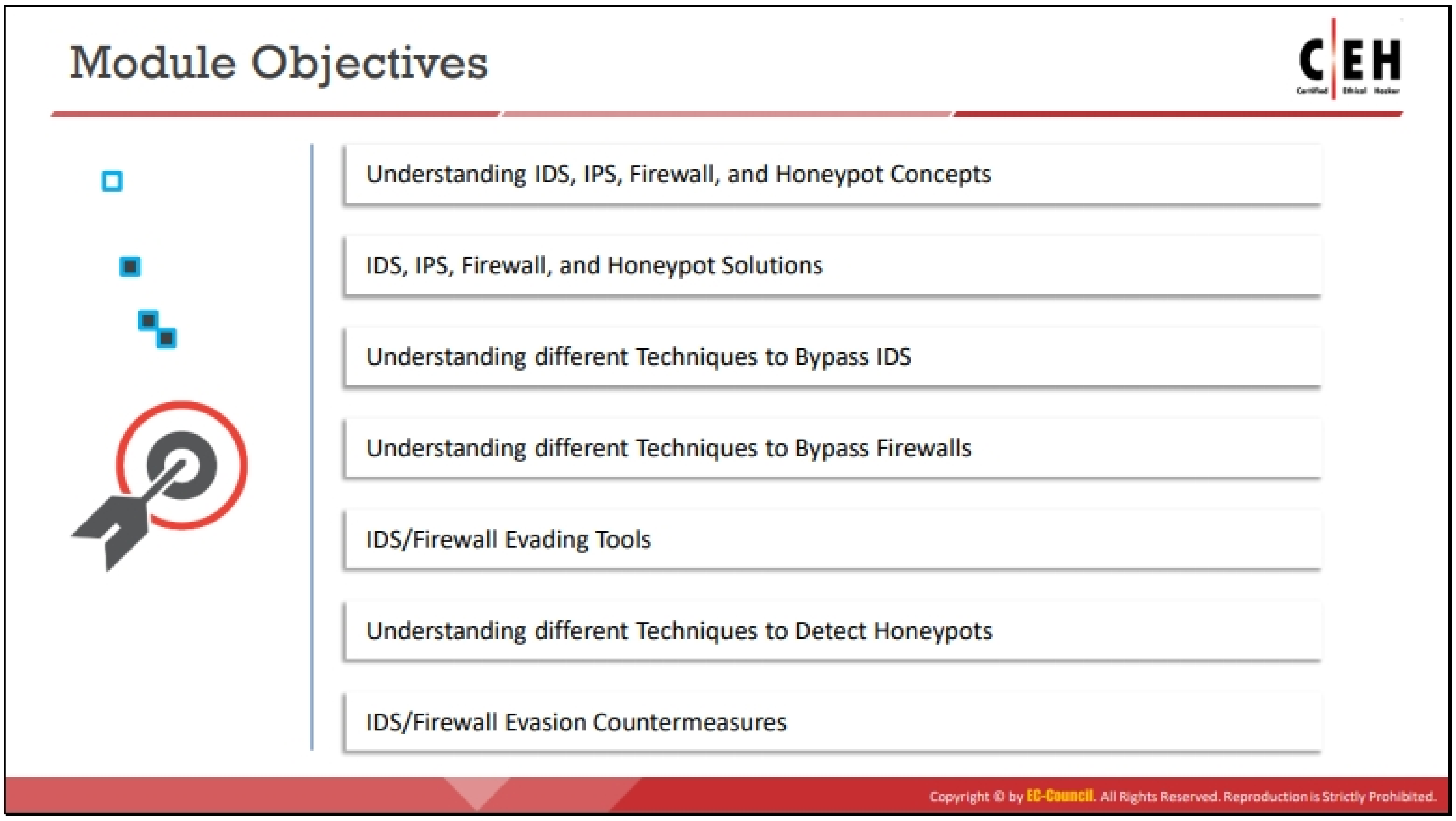
C|EH™

Certified Ethical Hacker

Module 12:

Evading IDS, Firewalls, and Honeypots



Module Objectives

- Understanding IDS, IPS, Firewall, and Honeypot Concepts
- IDS, IPS, Firewall, and Honeypot Solutions
- Understanding different Techniques to Bypass IDS
- Understanding different Techniques to Bypass Firewalls
- IDS/Firewall Evading Tools
- Understanding different Techniques to Detect Honeypots
- IDS/Firewall Evasion Countermeasures

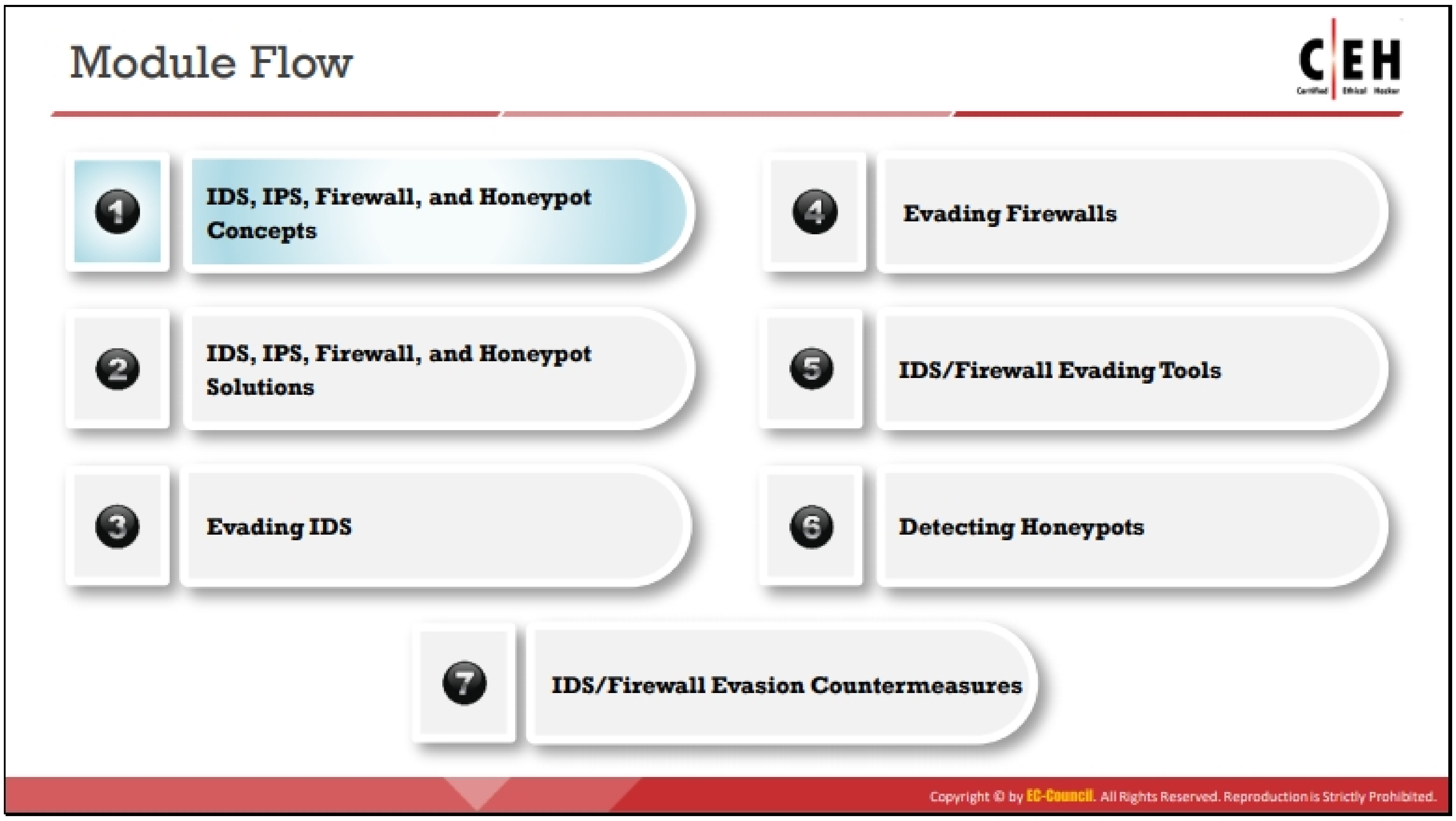
Module Objectives

The widespread use of the Internet throughout the business world has boosted network usage in general. Organizations adopt various network security measures such as firewalls, intrusion detection systems (IDS), intrusion prevention systems (IPS), and "honeypots" to protect their networks. Networks are the most preferred targets of hackers for compromising an organization's security, and attackers continue to find new ways to evade network security measures and attack these targets.

This module provides deep insights into various network security technologies, such as IDS, IPS, firewalls, and honeypots. It explains the operations of these components as well as the various techniques used by attackers to evade them. Further, it describes the countermeasures necessary to prevent such attacks.

At the end of this module, you will be able to:

- Describe IDS, IPS, firewall, and honeypot concepts
- Use different IDS, IPS, firewall, and honeypot solutions
- Explain different techniques to bypass IDS
- Explain various techniques to bypass firewalls
- Use different tools to evade IDS/firewalls
- Explain different techniques to detect honeypots
- Adopt countermeasures against IDS/firewall evasion

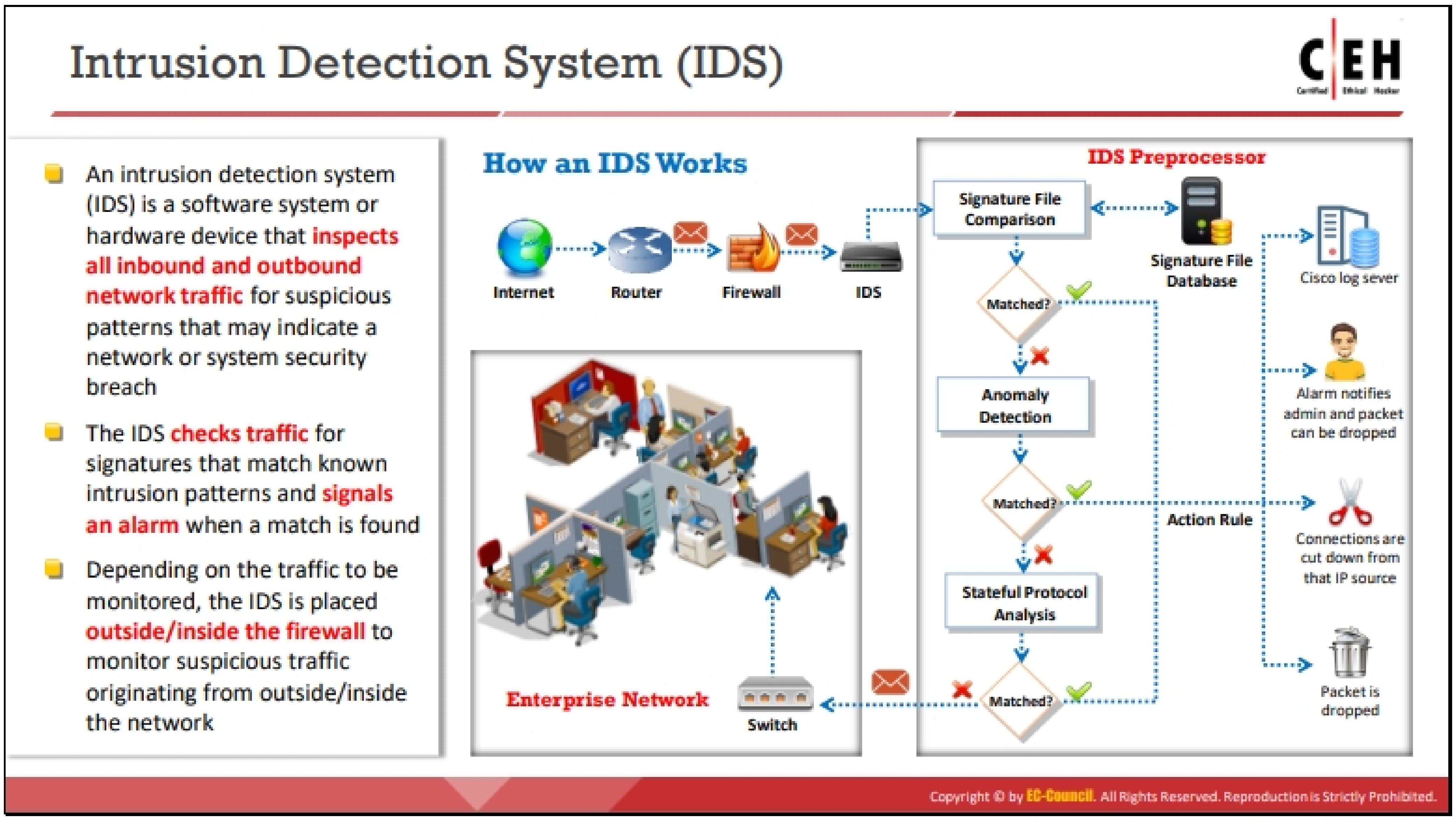


Module Flow

1. IDS, IPS, Firewall, and Honeypot Concepts
2. IDS, IPS, Firewall, and Honeypot Solutions
3. Evading IDS
4. Evading Firewalls
5. IDS/Firewall Evading Tools
6. Detecting Honeypots
7. IDS/Firewall Evasion Countermeasures

IDS, IPS, Firewall, and Honeypot Concepts

Ethical hackers should have an idea about the function, role, placement, and design of firewalls, IDS, IPS, and honeypots to protect an organization's network by understanding how an attacker evades such security measures. This section provides an overview of these basic concepts.



Intrusion Detection System (IDS)

- An intrusion detection system (IDS) is a software system or hardware device that **inspects all inbound and outbound network traffic** for suspicious patterns that may indicate a network or system security breach
- The IDS **checks traffic** for signatures that match known intrusion patterns and **signals an alarm** when a match is found
- Depending on the traffic to be monitored, the IDS is placed **outside/inside the firewall** to monitor suspicious traffic originating from outside/inside the network

How an IDS Works

Internet
Router
Firewall
IDS
IDS Preprocessor
Signature File Comparison
Signature File Database
Matched?
Anomaly Detection
Matched?
Action Rule
Stateful Protocol Analysis
Matched?
Cisco log sever
Alarm notifies admin and packet can be dropped
Connections are cut down from that IP source
Packet is dropped
Enterprise Network
Switch

Intrusion Detection System (IDS)

An intrusion detection system (IDS) is a security software or hardware device used to monitor, detect, and protect networks or systems from malicious activities; it alerts the concerned security personnel immediately upon detecting intrusions. IDS are extremely useful as they monitor the inbound/outbound traffic of the network and check for suspicious activities continuously to detect a network or system security breach. Specifically, they check traffic for signatures that match known intrusion patterns and raise an alarm when a match is detected. IDS can be categorized into active and passive IDS depending on their functionality. A passive IDS generally only detects intrusions while an active IPS not only detects intrusions in the network but also prevents them.

Main Functions of IDS:

- An IDS gathers and analyzes information from within a computer or a network to identify possible violations of the security policy, including unauthorized access, as well as misuse.
- An IDS is also referred to as a "packet sniffer," which intercepts packets traveling via various communication media and protocols, usually TCP/IP.
- The packets are analyzed after they are captured.
- An IDS evaluates traffic for suspected intrusions and raises an alarm upon detecting such intrusions.

Where IDS resides in the network

One of the most common places to deploy an IDS is near the firewall. Depending on the traffic to be monitored, an IDS is placed outside/inside the firewall to monitor suspicious traffic originating from outside/inside the network. When placed inside, the IDS will be ideal if it is near a DMZ; however, the best practice is to use a layered defense by deploying one IDS in front of the firewall and another one behind the firewall in the network.

Before deploying the IDS, it is essential to analyze the network topology, understand how the traffic flows to and from the resources that an attacker can use to gain access to the network, and identify the critical components that will be possible targets of various attacks against the network. After the position of the IDS in the network is determined, the IDS must be configured to maximize its network protection effect.

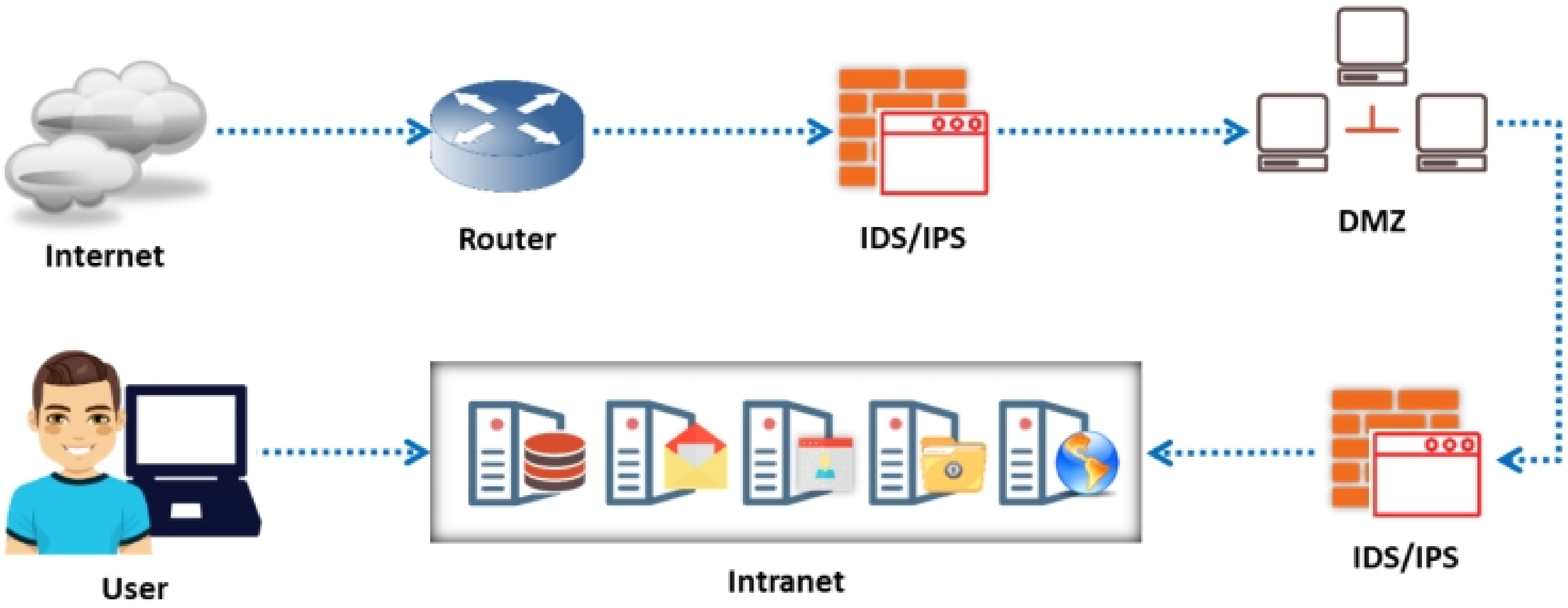


Figure 12.1: Placement of IDS

How an IDS Works

The primary purpose of the IDS is to provide real-time monitoring and detection of intrusions. Additionally, reactive IDS (and IPS) can intercept, respond to, and/or prevent intrusions.

An IDS works as follows:

- IDS have sensors to detect malicious signatures in data packets, and some advanced IDS include behavioral activity detection to detect malicious traffic behavior. Even if the packet signatures do not match perfectly with the signatures in the IDS signature database, the activity detection system can alert administrators about possible attacks.
- If the signature matches, the IDS performs predefined actions such as terminating the connection, blocking the IP address, dropping the packet, and/or raising an alarm to notify the administrator.
- When signature matches, anomaly detection will be skipped; otherwise, the sensor may analyze traffic patterns for an anomaly.
- When the packet passes all the tests, the IDS will forward it to the network.

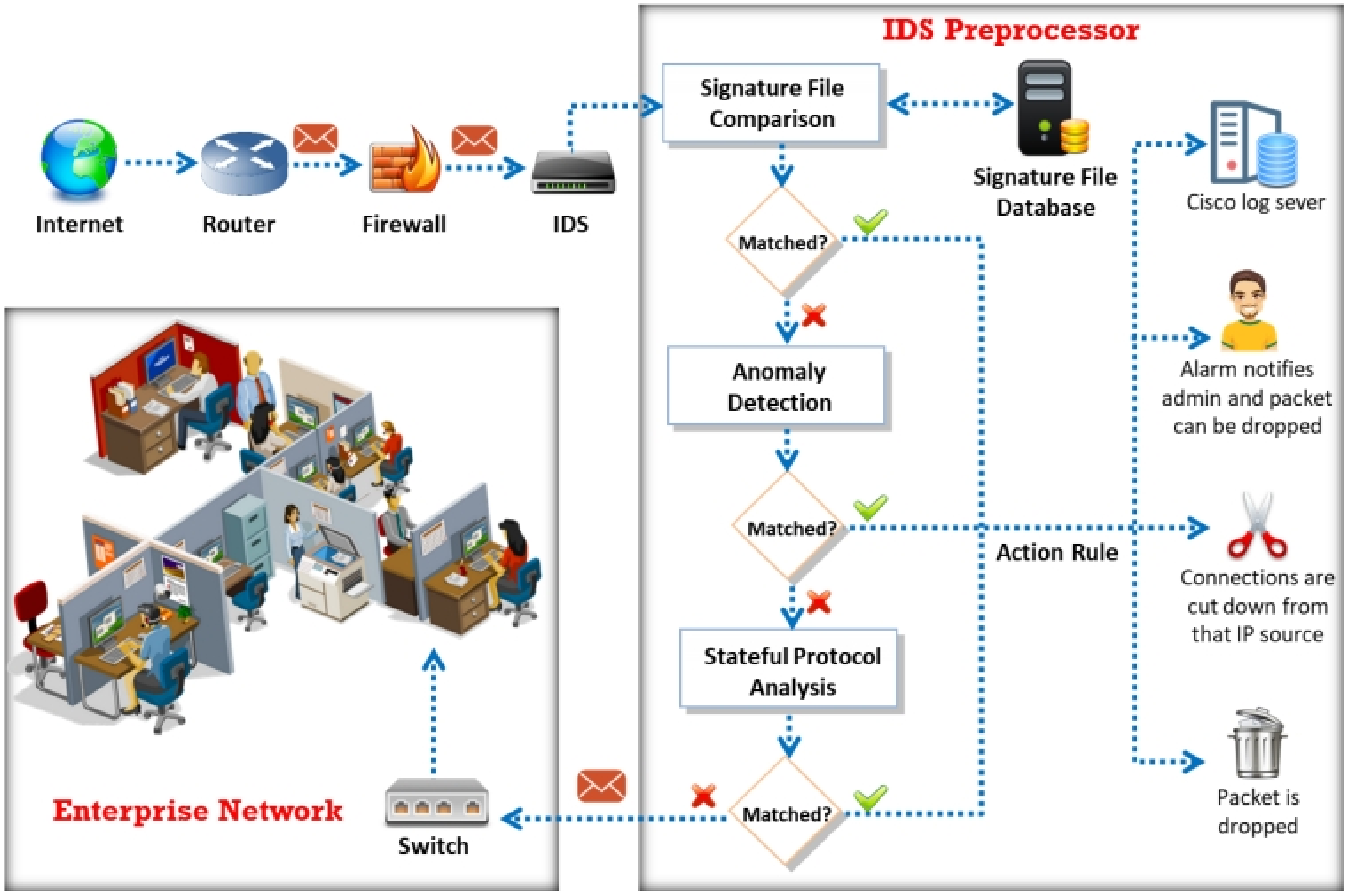
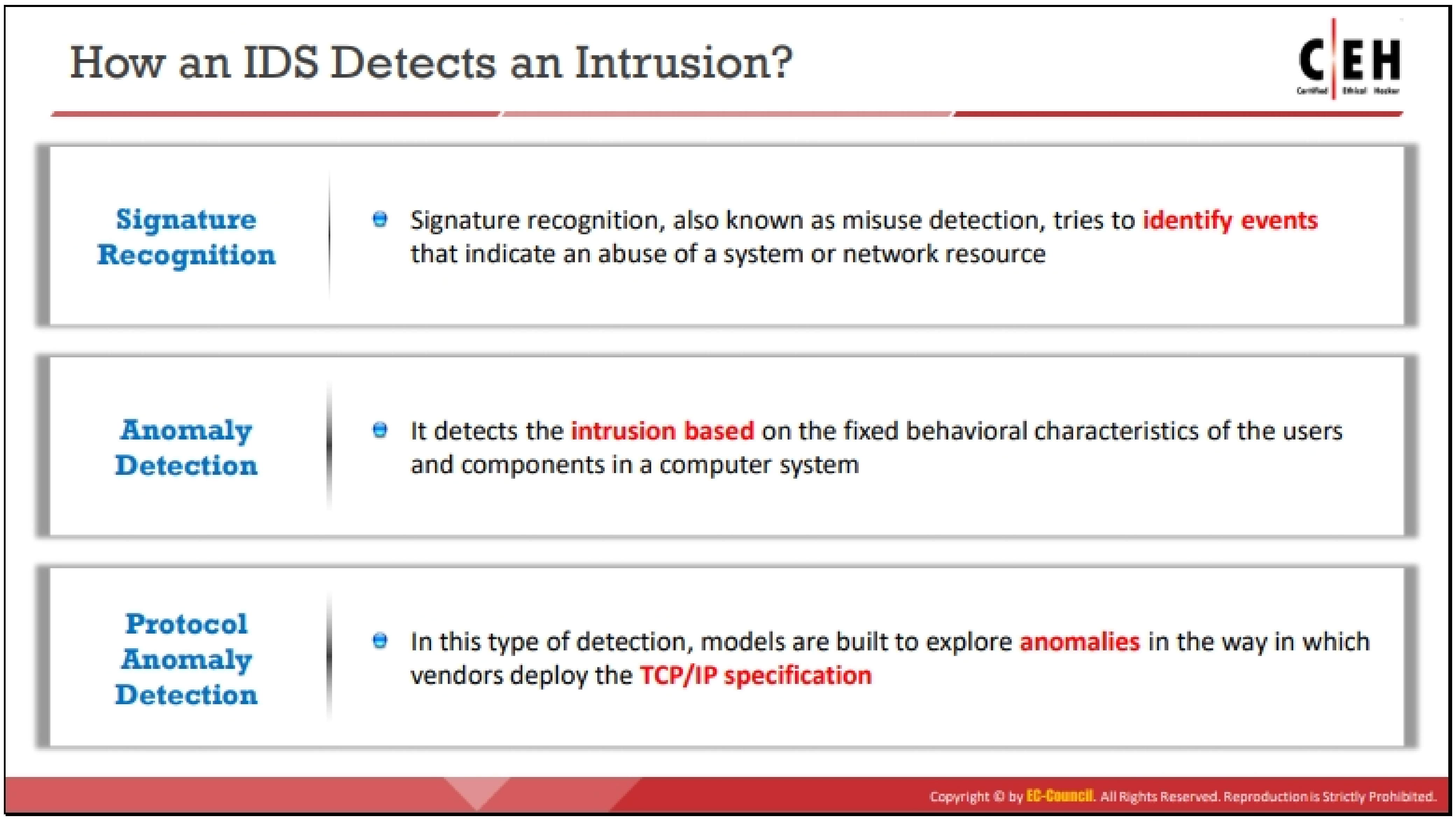

IDS Preprocessor
Internet
Router
Firewall
IDS
Signature File Comparison
Signature File Database
Cisco log sever
Matched?
Anomaly Detection
Alarm notifies admin and packet can be dropped
Matched?
Action Rule
Connections are cut down from that IP source
Stateful Protocol Analysis
Packet is dropped
Matched?
Enterprise Network
Switch

Figure 12.2: Working of IDS



How an IDS Detects an Intrusion?

Signature Recognition	- Signature recognition, also known as misuse detection, tries to **identify events** that indicate an abuse of a system or network resource
Anomaly Detection	- It detects the **intrusion based** on the fixed behavioral characteristics of the users and components in a computer system
Protocol Anomaly Detection	- In this type of detection, models are built to explore **anomalies** in the way in which vendors deploy the **TCP/IP specification**

How an IDS Detects an Intrusion?

An IDS uses three methods to detect intrusions in the network.

- **Signature Recognition**

  Signature recognition, also known as misuse detection, tries to identify events that indicate an abuse of a system or network. This technique involves first creating models of possible intrusions and then comparing these models with incoming events to make a detection decision. The signatures for IDS were created under the assumption that the model must detect an attack without disturbing normal system traffic. Only attacks should match the model; otherwise, false alarms could occur.

  - Signature-based intrusion detection compares incoming or outgoing network packets with the binary signatures of known attacks using simple pattern-matching techniques to detect intrusions. Attackers can define a binary signature for a specific portion of the packet, such as TCP flags.
  - Signature recognition can detect known attacks. However, there is a possibility that other innocuous packets contain the same signature, which will trigger a false positive alert.
  - Improper signatures may trigger false alerts. To detect misuse, a massive number of signatures are required. The more the signatures, the greater are the chances are of the IDS detecting attacks; however, the traffic may incorrectly match with the signatures, thus impeding system performance.
  - A large amount of signature data requires more network bandwidth. IDS compare signatures of data packets against those in the signature database. An increase in

the number of signatures in the database could result in the dropping of certain packets.

  - New virus attacks such as URSNIF and VIRLOCK have driven the need for multiple signatures for a single attack. Changing a single bit in some attack strings can invalidate a signature generated for that attack. Therefore, entirely new signatures are required to detect a similar attack.
  - Despite the problems with signature-based IDS, such systems are popular, and they work well when configured correctly and monitored closely.

- **Anomaly Detection**

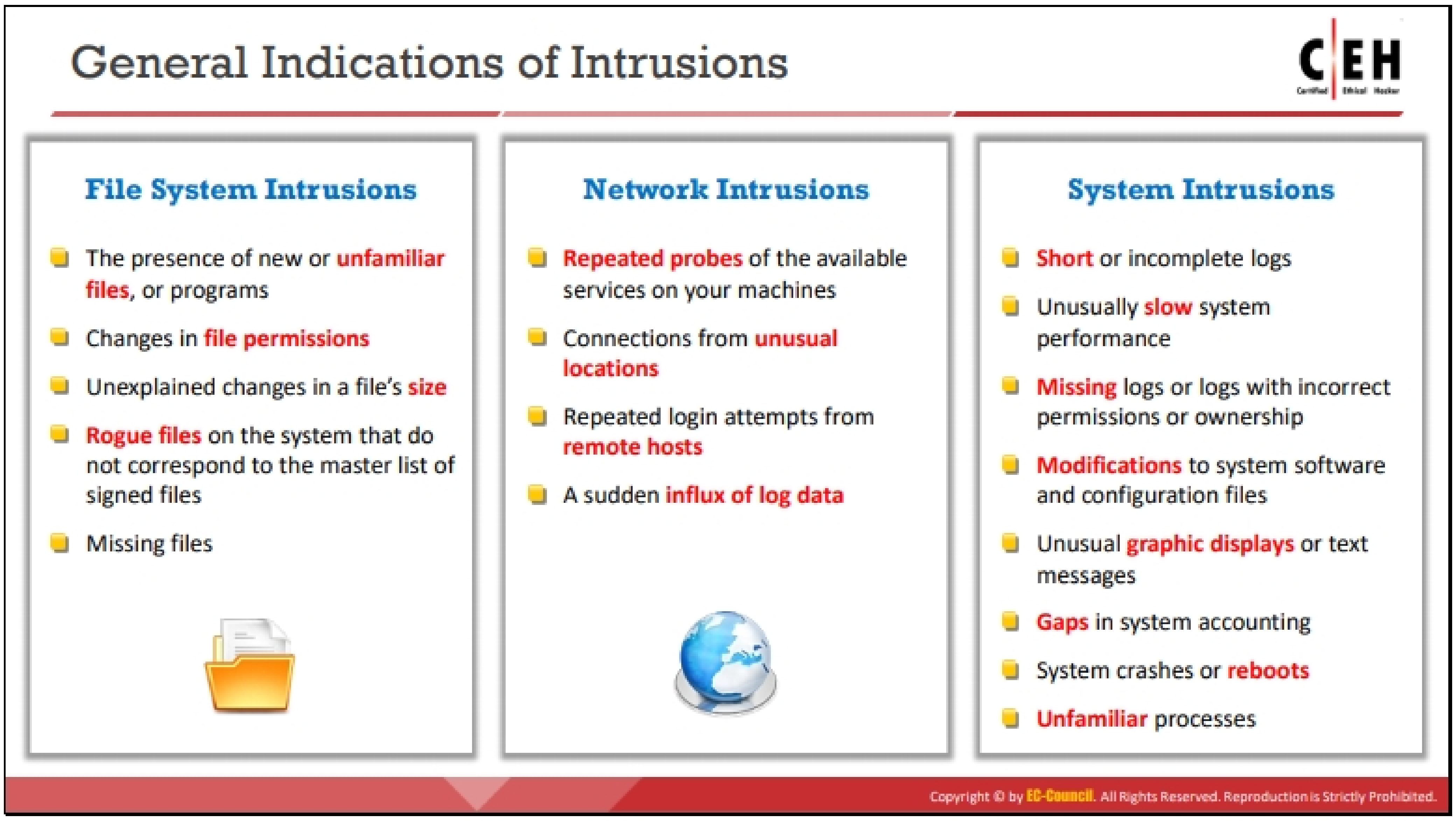
  Anomaly detection, or "**not-use detection**," differs from signature recognition. Anomaly detection involves a database of anomalies. An anomaly is detected when an event occurs outside the tolerance threshold of normal traffic. Therefore, any deviation from regular use is an attack. Anomaly detection detects intrusions based on the fixed behavioral characteristics of the users and components in a computer system. Establishing a model of normal use is the most challenging step in creating an anomaly detector.

  - In the traditional method of anomaly detection, essential data are kept for checking variations in network traffic. However, in reality, there is some unpredictability in network traffic, and there are too many statistical variations, thus making these models imprecise. Some events labeled as anomalies might only be irregularities in network usage.
  - In this type of approach, the inability to construct a model thoroughly on a regular network is a concern. These models should be used to check specific networks.

- **Protocol Anomaly Detection**

  Protocol anomaly detection depends on the anomalies specific to a protocol. It identifies particular flaws in vendors' deployment of the TCP/IP protocol. Protocols are designed according to RFC specifications, which dictate standard handshakes to permit universal communication. The protocol anomaly detector can identify new attacks.

  - There are new attack methods and exploits that violate protocol standards.
  - Malicious anomaly signatures are becoming increasingly common. By contrast, the network protocol is well defined and is changing slowly. Therefore, the signature database should frequently be updated to detect attacks.
  - Protocol anomaly detectors are different from traditional IDS in terms of how they present alarms.
  - The best way to present alarms is to explain which part of the state system is compromised. For this purpose, IDS operators must have thorough knowledge of protocol design.



General Indications of Intrusions

File System Intrusions

- The presence of new or **unfamiliar files**, or programs
- Changes in **file permissions**
- Unexplained changes in a file's **size**
- **Rogue files** on the system that do not correspond to the master list of signed files
- Missing files

Network Intrusions

- **Repeated probes** of the available services on your machines
- Connections from **unusual locations**
- Repeated login attempts from **remote hosts**
- A sudden **influx of log data**

System Intrusions

- **Short** or incomplete logs
- Unusually **slow** system performance
- **Missing** logs or logs with incorrect permissions or ownership
- **Modifications** to system software and configuration files
- Unusual **graphic displays** or text messages
- **Gaps** in system accounting
- System crashes or **reboots**
- **Unfamiliar** processes

General Indications of Intrusions

Intrusion attempts on networks, systems, or file systems can be identified by following some general indicators:

- **File System Intrusions**

  By observing system files, the presence of an intrusion can be identified. System files record the activities of the system. Any modification or deletion of the file attributes or the file itself is a sign that the system has been a target of an attack:

  - If you find new, unknown files/programs on your system, then there is a possibility that the system has been intruded into. The system can be compromised to the extent that it can, in turn, compromise other network systems.
  - When an intruder gains access to a system, he or she tries to escalate privileges to gain administrative access. When the intruder obtains administrator privileges, he/she could change file permissions, for example, from read-only to write.
  - Unexplained modifications in file size are also an indication of an attack. Make sure you analyze all your system files.
  - The presence of rogue suid and sgid files on your Linux system that do not match your master list of suid and sgid files could indicate an attack.
  - You can identify unfamiliar file names in directories, including executable files with strange extensions and double extensions.
  - Missing files are also a sign of a probable intrusion/attack.

- **Network Intrusions**

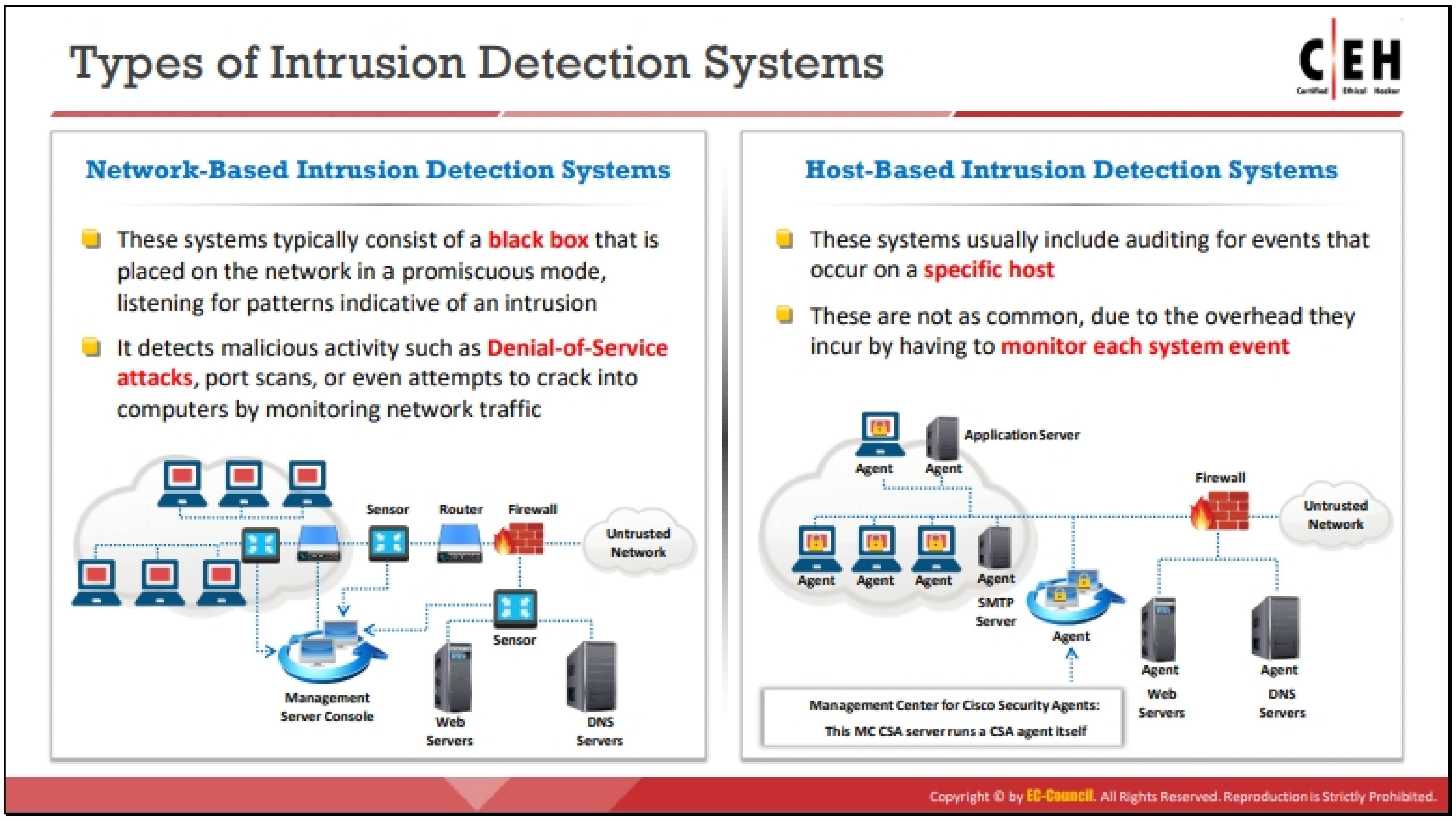
  Similarly, general indications of network intrusions include:

  - A sudden increase in bandwidth consumption
  - Repeated probes of the available services on your machines
  - Connection requests from IPs other than those in the network range, which imply that an unauthenticated user (intruder) is attempting to connect to the network
  - Repeated login attempts from remote hosts
  - A sudden influx of log data, which could indicate attempts at DoS attacks, bandwidth consumption, and DDoS attacks

- **System Intrusions**

  Similarly, general indications of system intrusions include:

  - Sudden changes in logs such as short or incomplete logs
  - Unusually slow system performance
  - Missing logs or logs with incorrect permissions or ownership
  - Modifications to system software and configuration files
  - Unusual graphic displays or text messages
  - Gaps in system accounting
  - System crashes or reboots
  - Unfamiliar processes



Types of Intrusion Detection Systems

Network-Based Intrusion Detection Systems

- These systems typically consist of a **black box** that is placed on the network in a promiscuous mode, listening for patterns indicative of an intrusion
- It detects malicious activity such as **Denial-of-Service attacks**, port scans, or even attempts to crack into computers by monitoring network traffic

Sensor
Router
Firewall
Untrusted Network
Sensor
Management Server Console
Web Servers
DNS Servers

Host-Based Intrusion Detection Systems

- These systems usually include auditing for events that occur on a **specific host**
- These are not as common, due to the overhead they incur by having to **monitor each system event**

Application Server
Agent
Agent
Firewall
Untrusted Network
Agent
Agent
Agent
Agent
SMTP Server
Agent
Agent
Web Servers
Agent
DNS Servers
Management Center for Cisco Security Agents:
This MC CSA server runs a CSA agent itself

Types of Intrusion Detection Systems

There are two types of intrusion detection systems:

- **Network-Based Intrusion Detection Systems**

  Network-based intrusion detection systems (NIDS) check every packet entering the network for the presence of anomalies and incorrect data. By limiting the firewall to drop large numbers of data packets, the NIDS checks every packet thoroughly. A NIDS captures and inspects all traffic. It generates alerts at the IP or application level based on the content. NIDS are more distributed than host-based IDS. The NIDS identifies the anomalies at the router and host levels. It audits the information contained in the data packets and logs the information of malicious packets; furthermore, it assigns a threat level to each risk after receiving the data packets. The threat level enables the security team to remain on alert. These mechanisms typically consist of a black box placed on the network in a promiscuous mode, listening for patterns indicative of an intrusion. It detects malicious activity such as DoS attacks, port scans, or even attempts to break into computers by monitoring network traffic.

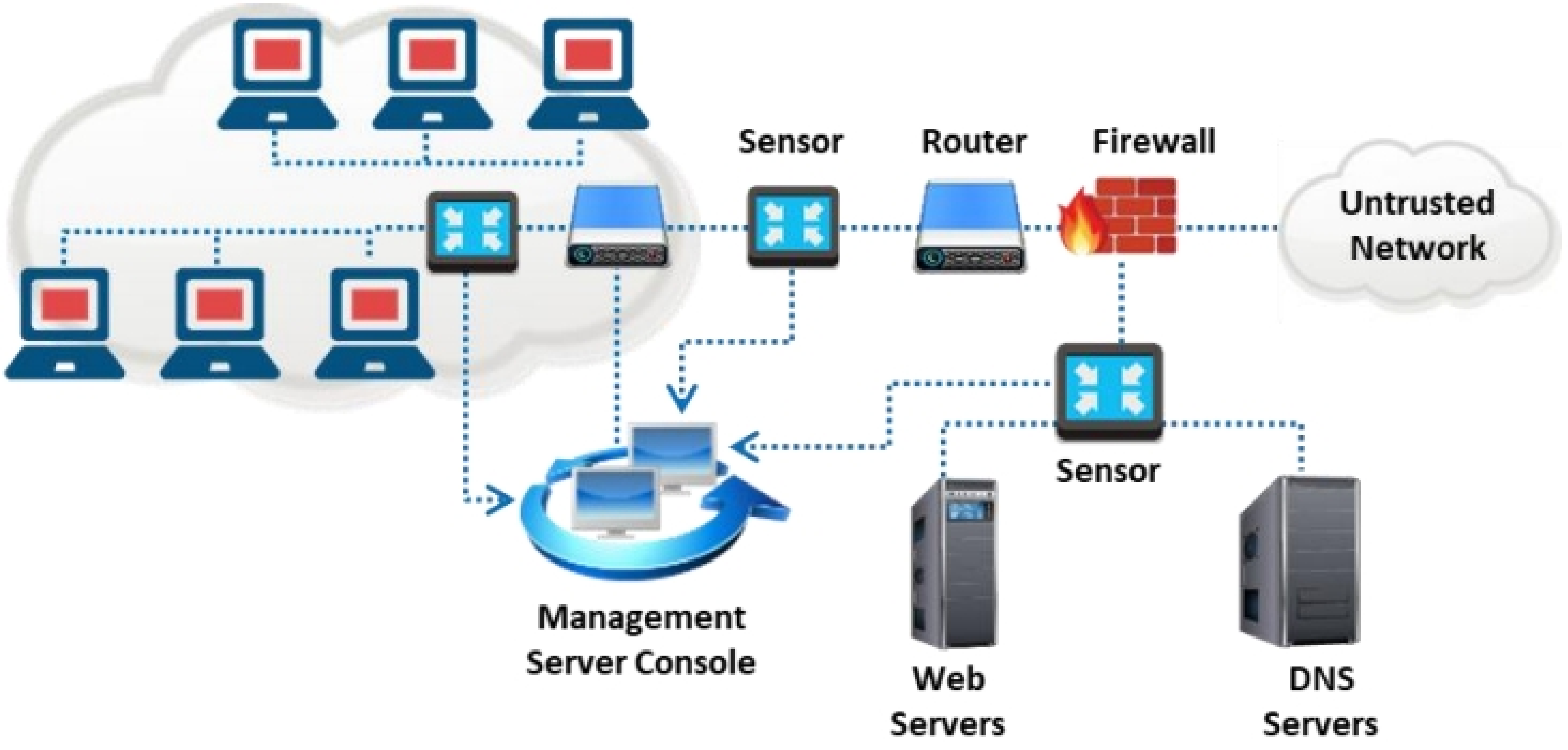

Sensor
Router
Firewall
Untrusted Network
Sensor
Management Server Console
Web Servers
DNS Servers

Figure 12.3: Network-based IDS (NIDS)

- **Host-Based Intrusion Detection Systems**

  A host-based IDS (HIDS) analyzes each system's behavior. The HIDS can be installed on any system ranging from a desktop PC to a server. It is more versatile than the NIDS. In addition to detecting unauthorized insider activity, host-based systems are also effective in detecting unauthorized file modification. The HIDS focuses on the changing aspects of local systems. It is also more platform-centric, with a greater focus on the Windows OS; nevertheless, other HIDS are available for UNIX platforms. These mechanisms usually include auditing events that occur on a specific host. They are not very common because of the overhead they incur by having to monitor each system event.

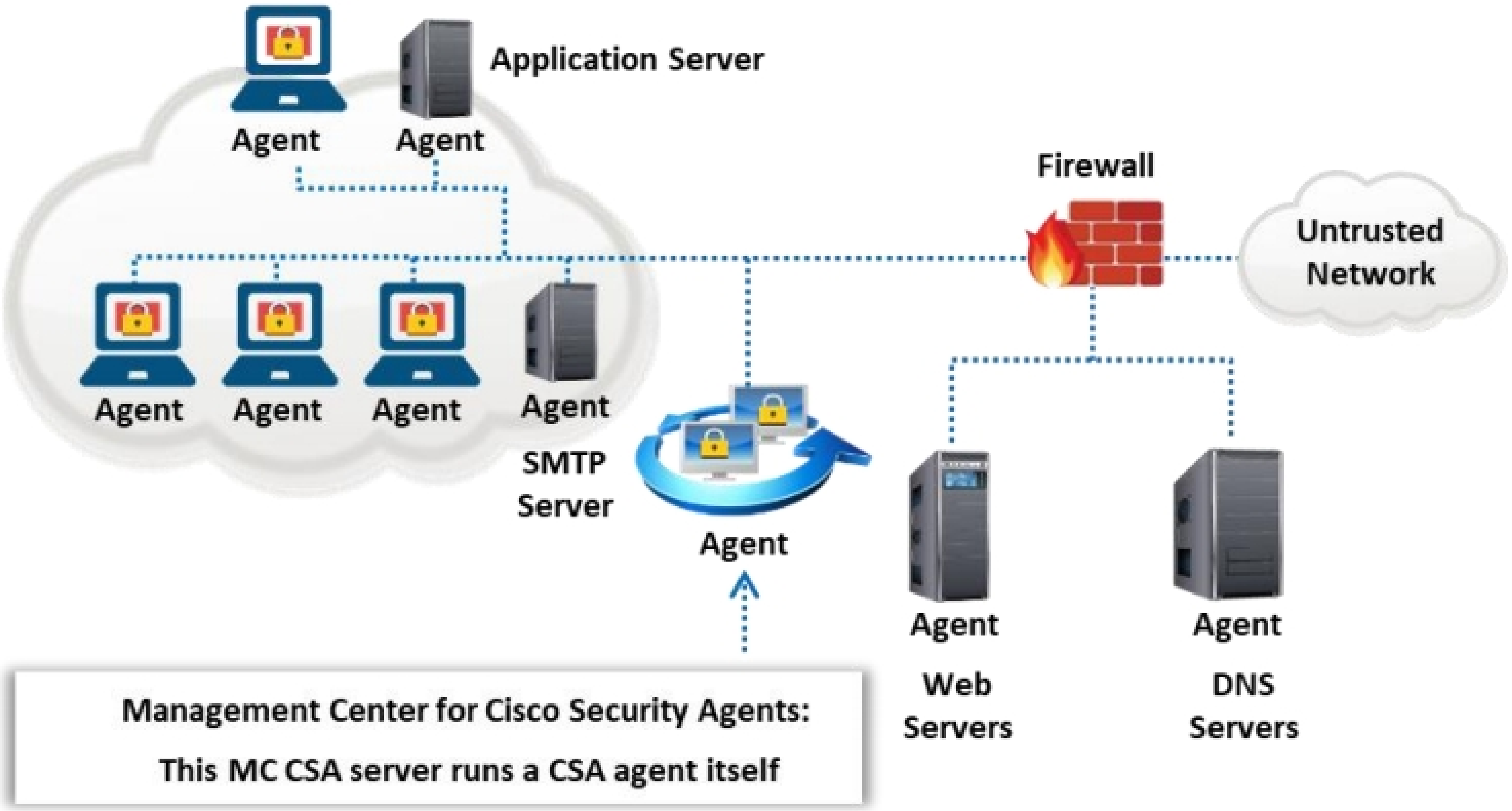
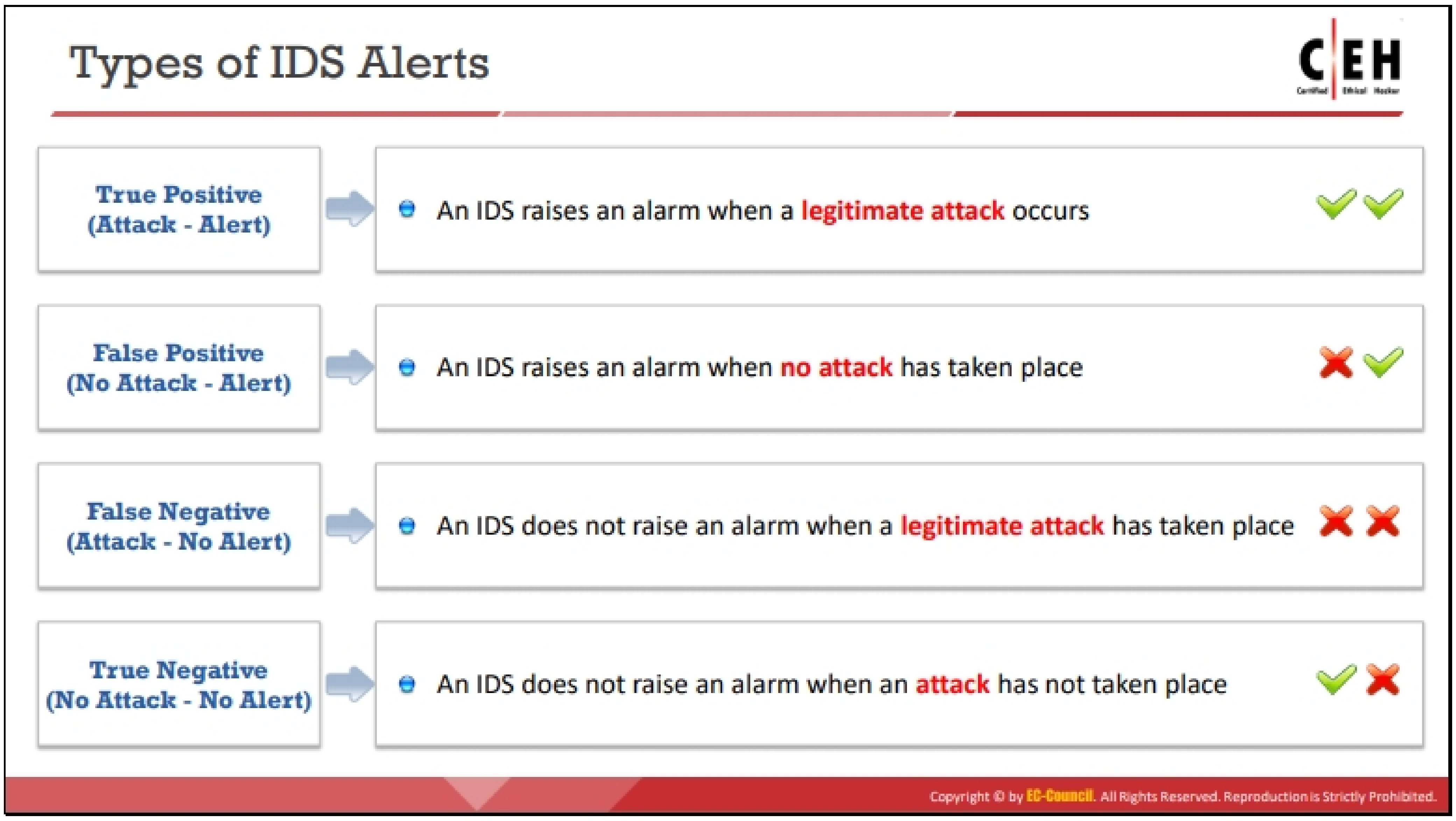

Application Server
Agent
Agent
Firewall
Untrusted Network
Agent
Agent
Agent
Agent
SMTP Server
Agent
Agent
Agent
Web Servers
DNS Servers
Management Center for Cisco Security Agents:
This MC CSA server runs a CSA agent itself

Figure 12.4: Host-based IDS (HIDS)



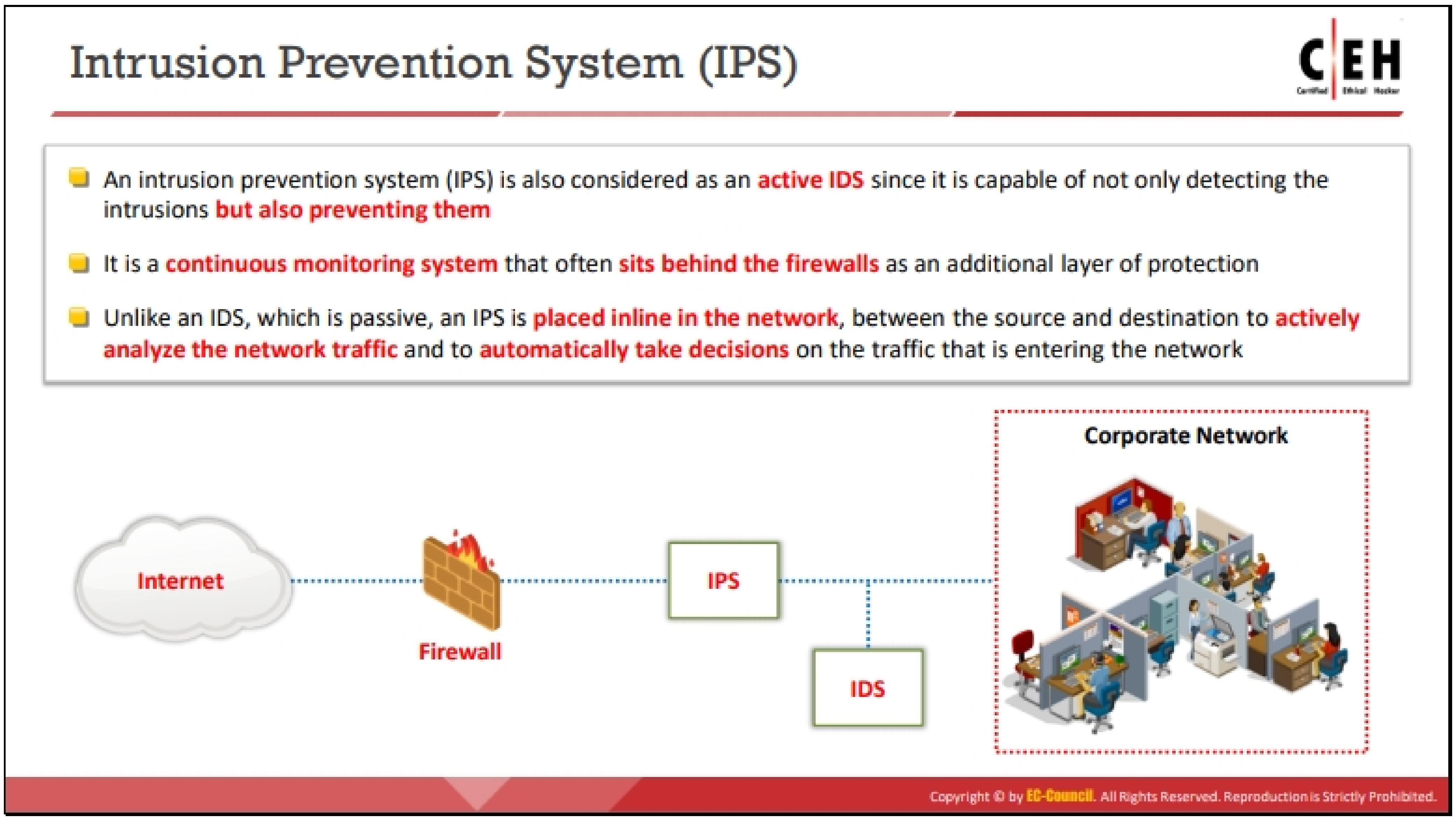
Types of IDS Alerts

True Positive (Attack - Alert)	An IDS raises an alarm when a **legitimate attack** occurs	✓✓
False Positive (No Attack - Alert)	An IDS raises an alarm when **no attack** has taken place	✗✓
False Negative (Attack - No Alert)	An IDS does not raise an alarm when a **legitimate attack** has taken place	✗✗
True Negative (No Attack - No Alert)	An IDS does not raise an alarm when an **attack** has not taken place	✓✗

Types of IDS Alerts

An IDS generates four types of alerts: True Positive, False Positive, False Negative, and True Negative.

- **True Positive (Attack - Alert)**: A true positive is a condition that occurs when an event triggers an alarm and causes the IDS to react as if a real attack is in progress. The event may be an actual attack, in which case an attacker attempts to compromise the network, or it may be a drill, in which case security personnel use hacker tools to test a network segment.
- **False Positive (No attack - Alert)**: A false positive occurs if an event triggers an alarm when no actual attack is in progress. It occurs when an IDS treats regular system activity as an attack. False positives tend to make users insensitive to alarms and weaken their reactions to actual intrusion events. While testing the configuration of an IDS, administrators use false positives to determine whether the IDS can distinguish between false positives and real attacks.
- **False Negative (Attack - No Alert)**: A false negative is a condition that occurs when an IDS fails to react to an actual attack event. This condition is the most dangerous failure, as the purpose of an IDS is to detect and respond to attacks.
- **True Negative (No attack - No Alert)**: A true negative is a condition that occurs when an IDS identifies an activity as acceptable behavior and the activity is acceptable. A true negative means successfully ignoring acceptable behavior. It is not harmful, as the IDS performs as expected in this case.



Intrusion Prevention System (IPS)

- An intrusion prevention system (IPS) is also considered as an **active IDS** since it is capable of not only detecting the intrusions **but also preventing them**
- It is a **continuous monitoring system** that often **sits behind the firewalls** as an additional layer of protection
- Unlike an IDS, which is passive, an IPS is **placed inline in the network**, between the source and destination to **actively analyze the network traffic** and to **automatically take decisions** on the traffic that is entering the network

Intrusion Prevention System (IPS)

Intrusion prevention systems (IPS) are considered as active IDS, as they are capable of not only detecting intrusions but also preventing them. IPS are continuous monitoring systems that often sit behind firewalls as an additional layer of protection. Unlike IDS, which are passive, IPS are placed inline in the network, between the source and the destination, to actively analyze the network traffic and make automated decisions regarding the traffic that is entering the network.

Some of the actions that an IPS is meant to perform are as follows:

- Generate alerts if any abnormal traffic is detected in the network
- Continuously record real-time logs of network activities
- Block and filter malicious traffic
- Detect and eliminate threats quickly, as it is placed inline in the operational network
- Identify threats accurately without generating false positives

An IPS takes actions based on certain rules and policies configured into it. In other words, the IPS can identify, log, and prevent the occurrence of any intrusion or attack in the network. IPS can also be employed to detect critical issues in corporate security policies such as notorious insider threats, malicious network guests, etc.

Classification of IPS:

Like IDS, IPS are also classified into two types:

- Host-based IPS
- Network-based IPS

Advantages of IPS over IDS:

- Unlike IDS, IPS can block as well as drop illegal packets in the network
- IPS can be used to monitor activities occurring in a single organization
- IPS can prevent the occurrence of direct attacks in the network by controlling the amount of network traffic

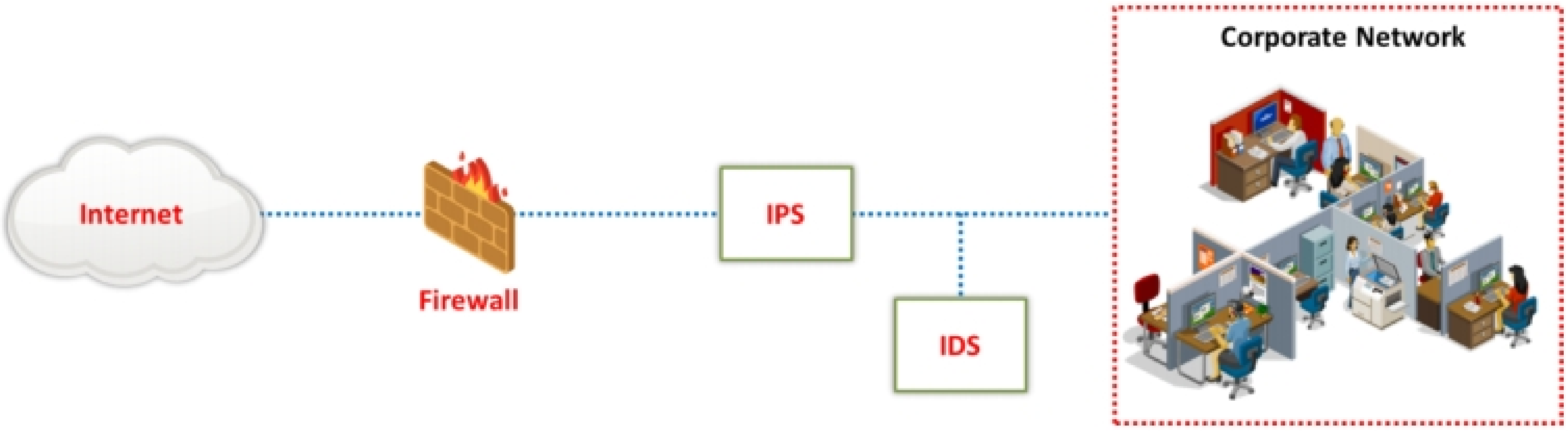
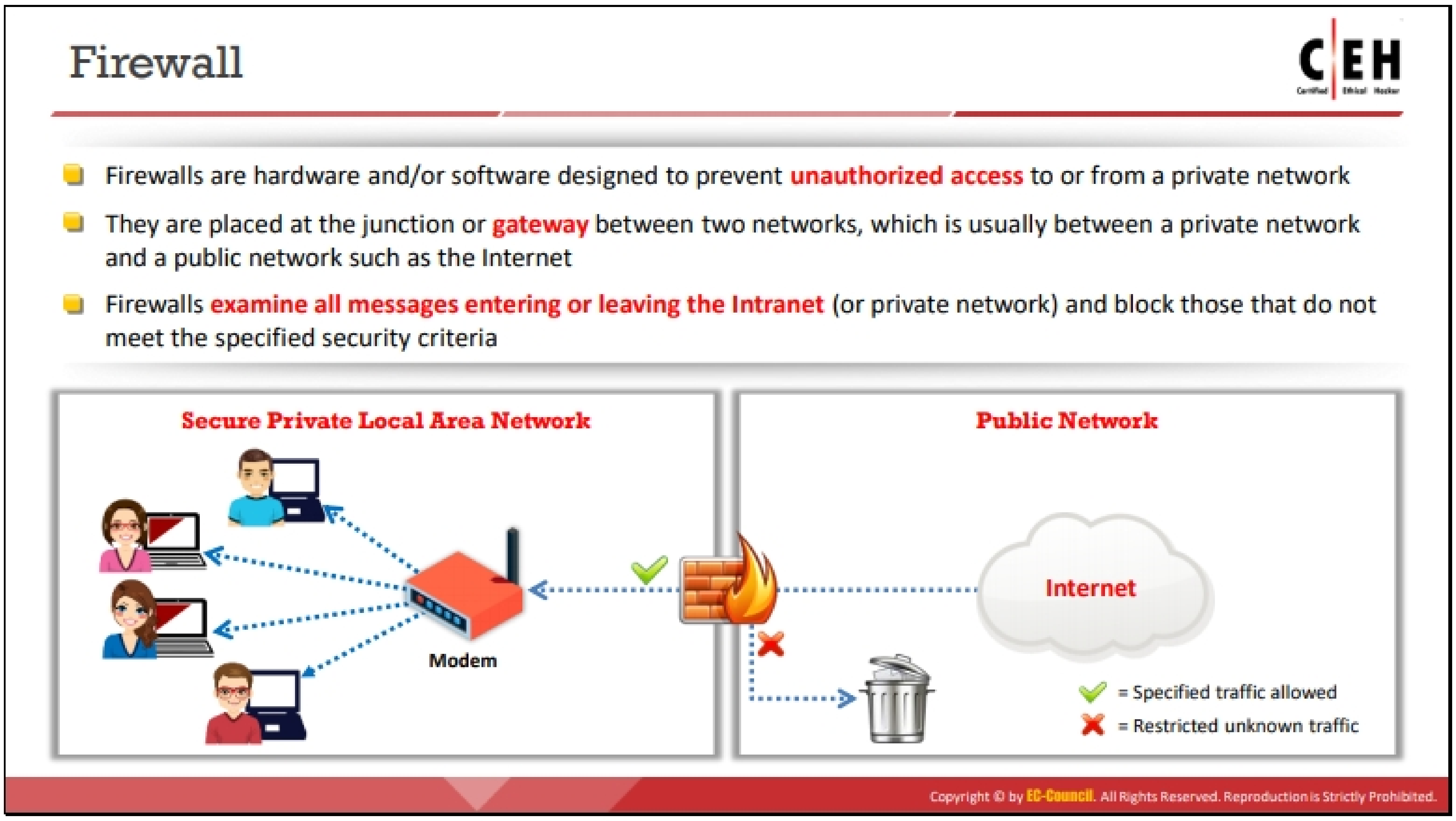


Figure 12.5: Example of an IPS placement



Firewall

- Firewalls are hardware and/or software designed to prevent **unauthorized access** to or from a private network
- They are placed at the junction or **gateway** between two networks, which is usually between a private network and a public network such as the Internet
- Firewalls **examine all messages entering or leaving the Intranet** (or private network) and block those that do not meet the specified security criteria

Secure Private Local Area Network
Public Network
Modem
Internet
✓ = Specified traffic allowed
✗ = Restricted unknown traffic

Firewall

A firewall is a software- or hardware-based system located at the network gateway that protects the resources of a private network from unauthorized access by users on other networks. They are placed at the junction or gateway between two networks, usually a private network and a public network such as the Internet. Firewalls examine all the messages entering or leaving the intranet and block those that do not meet the specified security criteria. Firewalls may be concerned with the type of traffic or with the source or destination addresses and ports. They include a set of tools that monitor the flow of traffic between networks. A firewall placed at the network level and working closely with the router filters all the network packets to determine whether to forward them toward their destinations. Always install firewalls away from the rest of the network, so that none of the incoming requests can gain direct access to a private network resource. If appropriately configured, the firewall protects systems on one side of it from systems on the other side.

- A firewall is an intrusion detection mechanism that is designed by an organization's security policy. Its settings can change to make appropriate changes to its functionality.
- Firewalls can be configured to restrict incoming traffic to POP and SMTP and to enable email access. Certain firewalls block specific email services to avoid spam.
- A firewall can be configured to check inbound traffic at a "checkpoint," where a security audit is performed. It can also act as an active "phone tap" tool for identifying an intruder's attempt to dial into modems in a secured network. Firewall logs consist of logging information that notifies the administrator about all attempts to access various services.

- The firewall verifies the incoming and outgoing traffic against its rules and acts as a router to move data between networks. The firewall allows or denies access requests made from one side of it to services on the other side.
- Identify all the attempts to log into the network for auditing. Unauthorized attempts can be identified by embedding an alarm that is triggered when an unauthorized user attempts to log in. Firewalls can filter packets based on the address and type of traffic. They recognize the source and destination addresses as well as port numbers during address filtering, and they identify the types of network traffic during protocol filtering. Firewalls can identify the state and attributes of data packets.

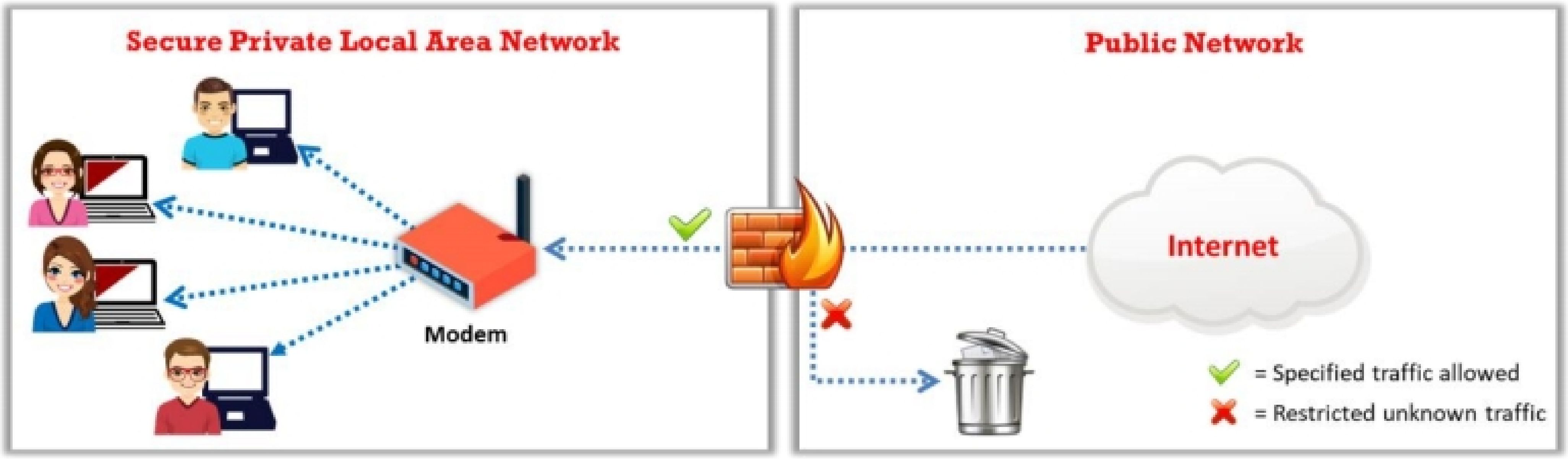
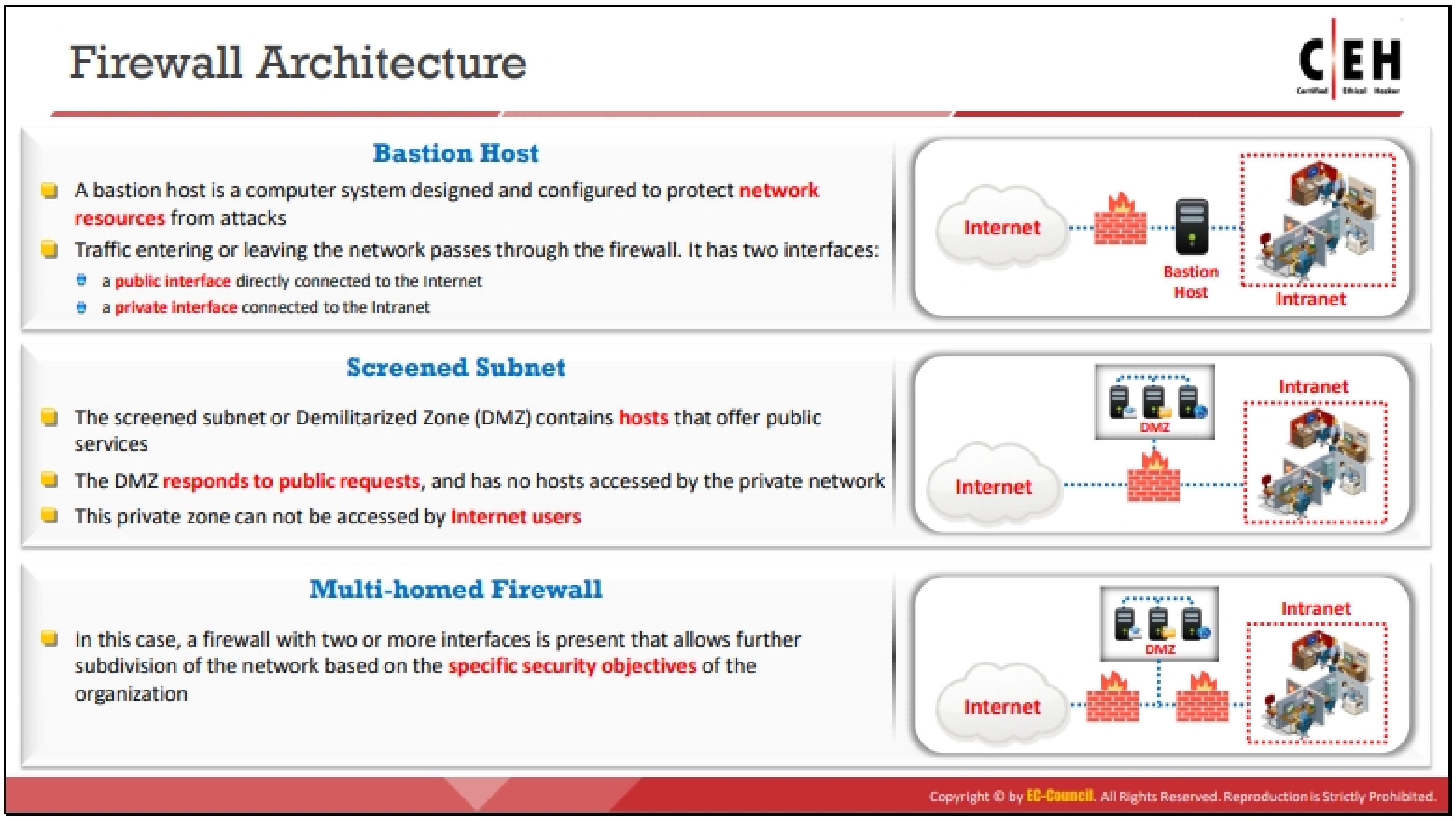

Secure Private Local Area Network
Modem
Public Network
Internet
✓ = Specified traffic allowed
✗ = Restricted unknown traffic

Figure 12.6: Example of a Firewall



Firewall Architecture

Bastion Host

- A bastion host is a computer system designed and configured to protect **network resources** from attacks
- Traffic entering or leaving the network passes through the firewall. It has two interfaces:
  - a **public interface** directly connected to the Internet
  - a **private interface** connected to the Intranet

Screened Subnet

- The screened subnet or Demilitarized Zone (DMZ) contains **hosts** that offer public services
- The DMZ **responds to public requests**, and has no hosts accessed by the private network
- This private zone can not be accessed by **Internet users**

Multi-homed Firewall

- In this case, a firewall with two or more interfaces is present that allows further subdivision of the network based on the **specific security objectives** of the organization

Firewall Architecture

The firewall architecture consists of the following elements:

- **Bastion Host**

  The bastion host is designed for defending the network against attacks. It acts as a mediator between inside and outside networks. A bastion host is a computer system designed and configured to protect network resources from attacks. Traffic entering or leaving the network passes through the firewall. It has two interfaces:

  - Public interface directly connected to the Internet
  - Private interface connected to the intranet

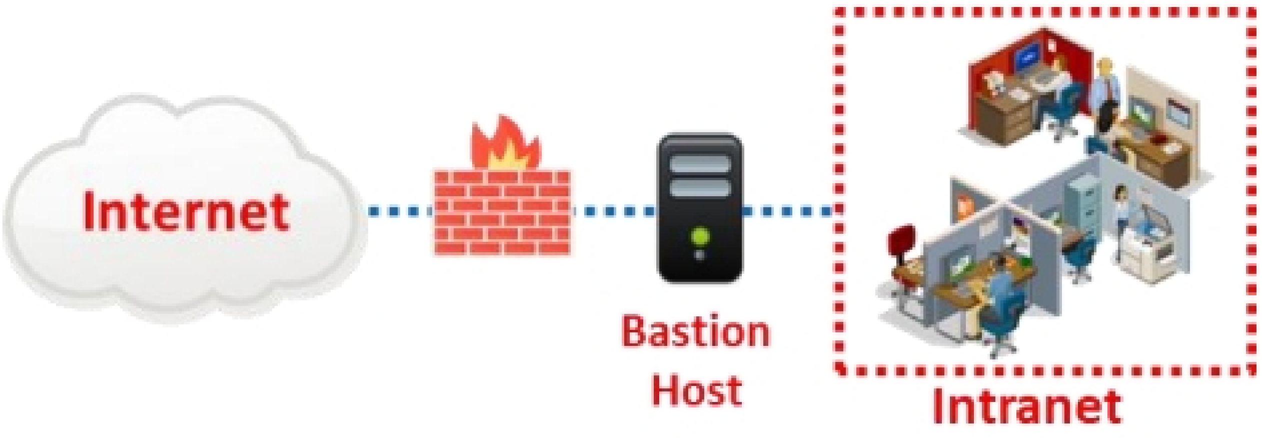


Figure 12.7: Bastion Host Firewall

- **Screened Subnet**

  A screened subnet (DMZ) is a protected network created with a two- or three-homed firewall behind a screening firewall, and it is a term that is commonly used to refer to the DMZ. When using a three-homed firewall, connect the first interface to the Internet, the second to the DMZ, and the third to the intranet. The DMZ responds to public requests and has no hosts accessed by the private network. Internet users cannot access the private zone.

The advantage of screening a subnet away from the intranet is that public requests can be responded to without allowing traffic into the intranet. A disadvantage of the three-homed firewall is that if it is compromised, both the DMZ and the intranet could also be compromised. A safer technique is to use multiple firewalls to separate the Internet from the DMZ, and to then separate the DMZ from the intranet.

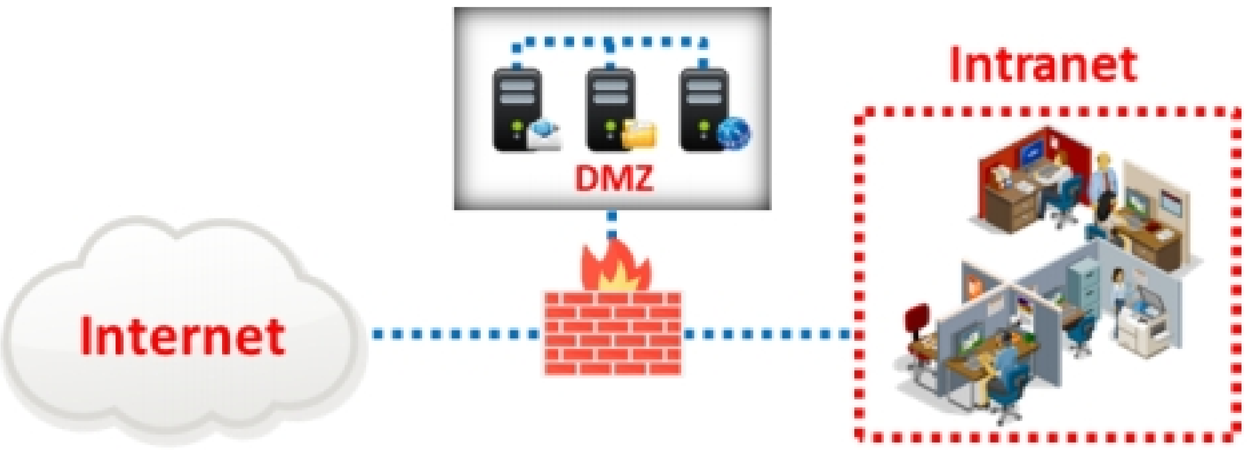


Figure 12.8: Screened Subnet Firewall

- **Multi-homed Firewall**

  A multi-homed firewall is a node with multiple NICs that connects to two or more networks. It connects each interface to separate network segments logically and physically. A multi-homed firewall helps in increasing the efficiency and reliability of an IP network. The multi-homed firewall has more than three interfaces that allow for further subdividing the systems based on the specific security objectives of the organization. However, the model that provides deeper protection is the back-to-back firewall.

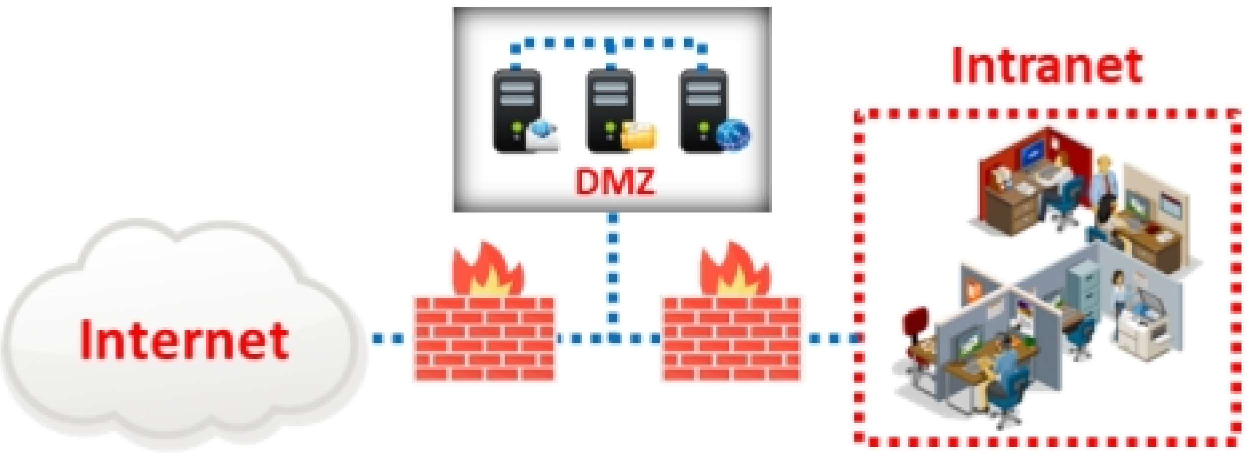
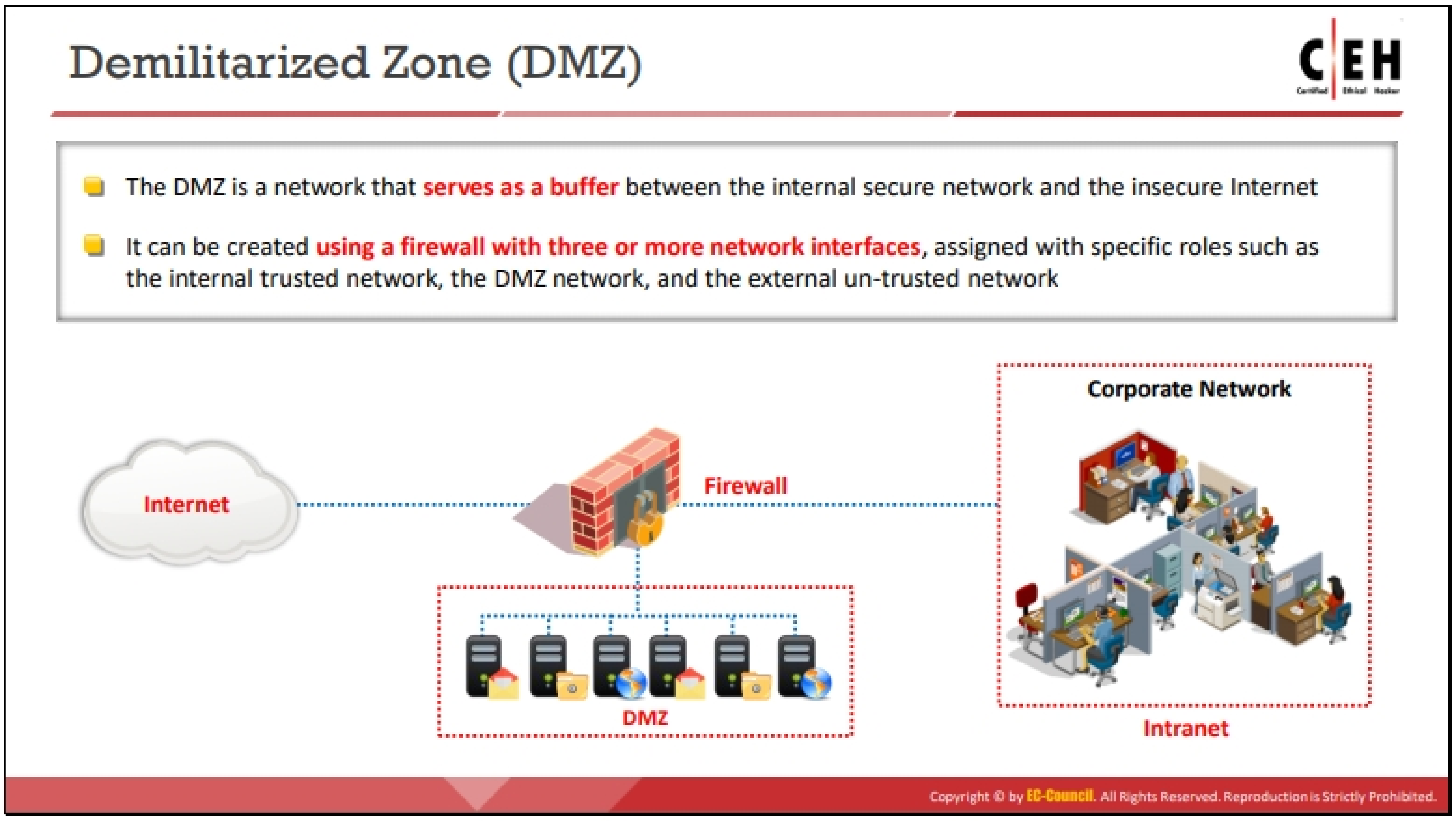


Figure 12.9: Multi-homed Firewall



Demilitarized Zone (DMZ)

- The DMZ is a network that **serves as a buffer** between the internal secure network and the insecure Internet
- It can be created **using a firewall with three or more network interfaces**, assigned with specific roles such as the internal trusted network, the DMZ network, and the external un-trusted network

Corporate Network
Internet
Firewall
DMZ
Intranet

Demilitarized Zone (DMZ)

In computer networks, the demilitarized zone (DMZ) is an area that hosts computer(s) or a small sub-network placed as a neutral zone between a particular company's internal network and an untrusted external network to prevent outsider access to a company's private data. The DMZ serves as a buffer between the secure internal network and the insecure Internet, as it adds a layer of security to the corporate LAN, thus preventing direct access to other parts of the network.

A DMZ is created using a firewall with three or more network interfaces that are assigned specific roles, such as an internal trusted network, a DMZ network, or an external untrusted network (Internet). Any service such as email, web, or FTP that provides access to external users can be placed in the DMZ. However, web servers that communicate with database servers cannot reside in the DMZ, as they could give outside users direct access to sensitive information. There are many ways in which the DMZ can be configured according to specific network topologies and company requirements.

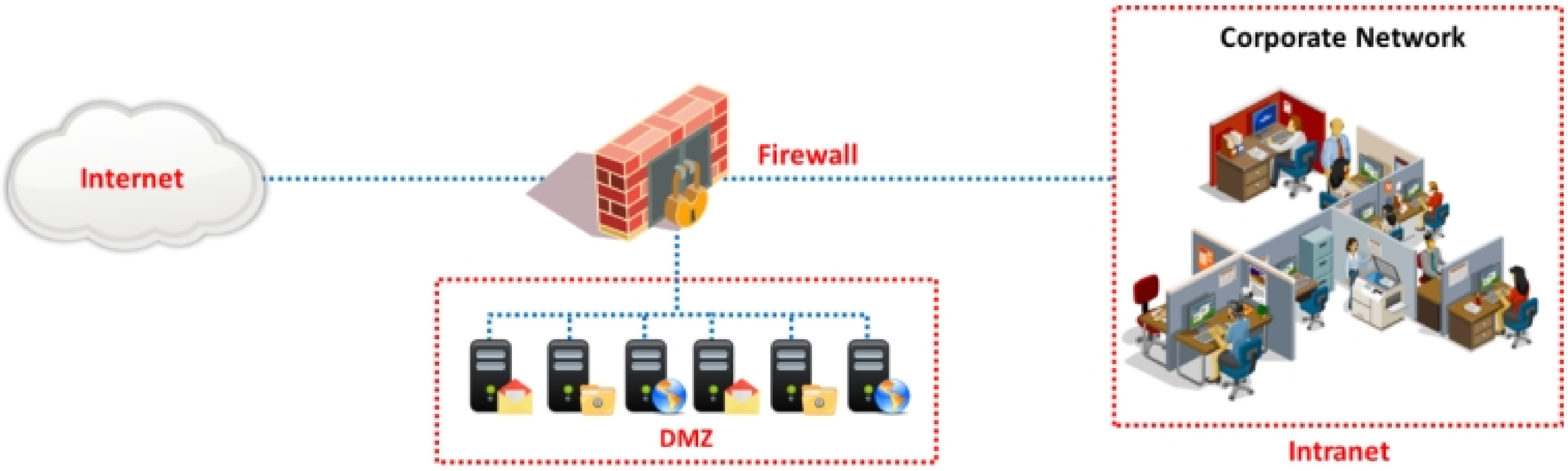
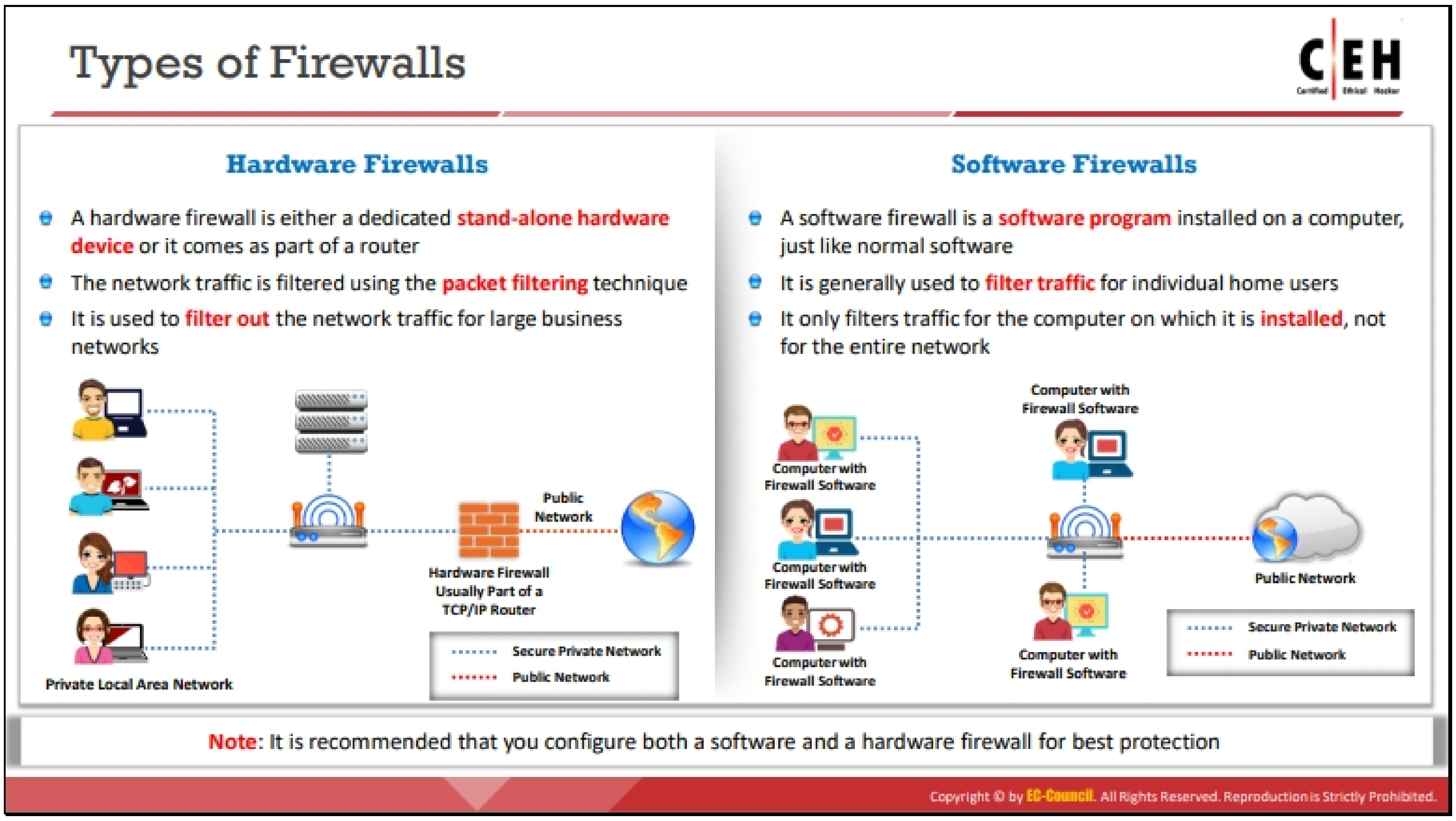


Figure 12.10: Demilitarized Zone (DMZ)



Types of Firewalls

Hardware Firewalls

- A hardware firewall is either a dedicated **stand-alone hardware device** or it comes as part of a router
- The network traffic is filtered using the **packet filtering** technique
- It is used to **filter out** the network traffic for large business networks

Public Network
Hardware Firewall Usually Part of a TCP/IP Router
Private Local Area Network
Secure Private Network
Public Network

Software Firewalls

- A software firewall is a **software program** installed on a computer, just like normal software
- It is generally used to **filter traffic** for individual home users
- It only filters traffic for the computer on which it is **installed**, not for the entire network

Computer with Firewall Software
Computer with Firewall Software
Computer with Firewall Software
Computer with Firewall Software
Computer with Firewall Software
Public Network
Secure Private Network
Public Network

Note: It is recommended that you configure both a software and a hardware firewall for best protection

Types of Firewalls

There are two types of firewalls.

- **Hardware Firewalls**

  A hardware firewall is a dedicated firewall device placed on the perimeter of the network. It is an integral part of the network setup and is also built into broadband routers or used as a standalone product. A hardware firewall helps to protect systems on the local network and performs effectively with little or no configuration. It employs the technique of packet filtering. It reads the header of a packet to find out the source and destination addresses, and compares them with a set of predefined and/or user-created rules that determine whether it should forward or drop the packet. A hardware firewall functions on an individual system or a particular network connected using a single interface. Examples of hardware firewalls include Cisco ASA and FortiGate. Hardware firewalls protect the private local area network.

  However, hardware firewalls are expensive as well as difficult to implement and upgrade.

  Advantages:

  - Security: A hardware firewall with its operating system (OS) is considered to reduce security risks and increase the level of security controls.
  - Speed: Hardware firewalls initiate faster responses and enable more traffic.
  - Minimal Interference: Since a hardware firewall is a separate network component, it enables better management and allows the firewall to shut down, move, or be reconfigured without much interference in the network.

Disadvantages:

- More expensive than a software firewall.
- Difficult to implement and configure.
- Consumes more space and involves cabling.

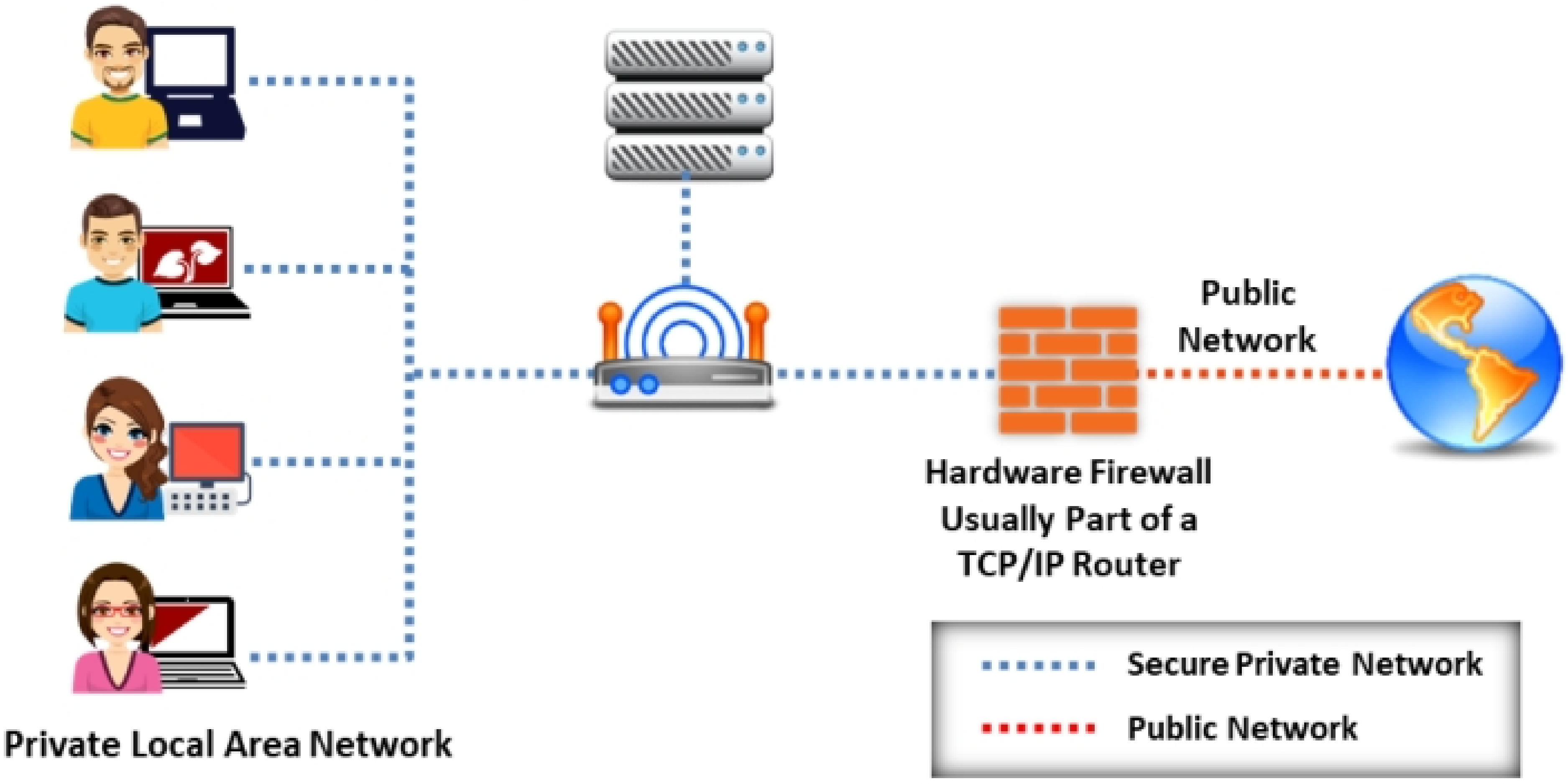

Public Network
Hardware Firewall Usually Part of a TCP/IP Router
Private Local Area Network
Secure Private Network
Public Network

Figure 12.11: Hardware Firewall

- **Software Firewalls**

  A software firewall is similar to a filter. It sits between a regular application and the networking components of the OS. It is more useful for individual home users and it is suitable for mobile users who need digital security when working outside the corporate network. Further, it is easy to install on an individual's PC, notebook, or workgroup server. It helps protect your system from outside attempts at unauthorized access and provides protection against everyday Trojans and email worms. It includes privacy controls, web filtering, and more. A software firewall implants itself in the critical area of the application/network path. It analyzes the data flow against the rule set.

  The configuration of a software firewall is simple compared to that of a hardware firewall. A software firewall intercepts all requests from a network to the computer to determine if they are valid and protects the computer from attacks and unauthorized access. It incorporates user-defined controls, privacy controls, web filtering, content filtering, etc., to restrict unsafe applications from running on an individual system. Software firewalls use more resources than hardware firewalls, which reduces the speed of the system. Examples of software firewalls include those produced by Norton, McAfee, and Kaspersky.

  Advantages:

  - Less expensive than hardware firewalls.
  - Ideal for personal or home use.

- Easier to configure and reconfigure.

Disadvantages:

- Consumes system resources.
- Difficult to uninstall.
- Not appropriate for environments requiring faster response times.

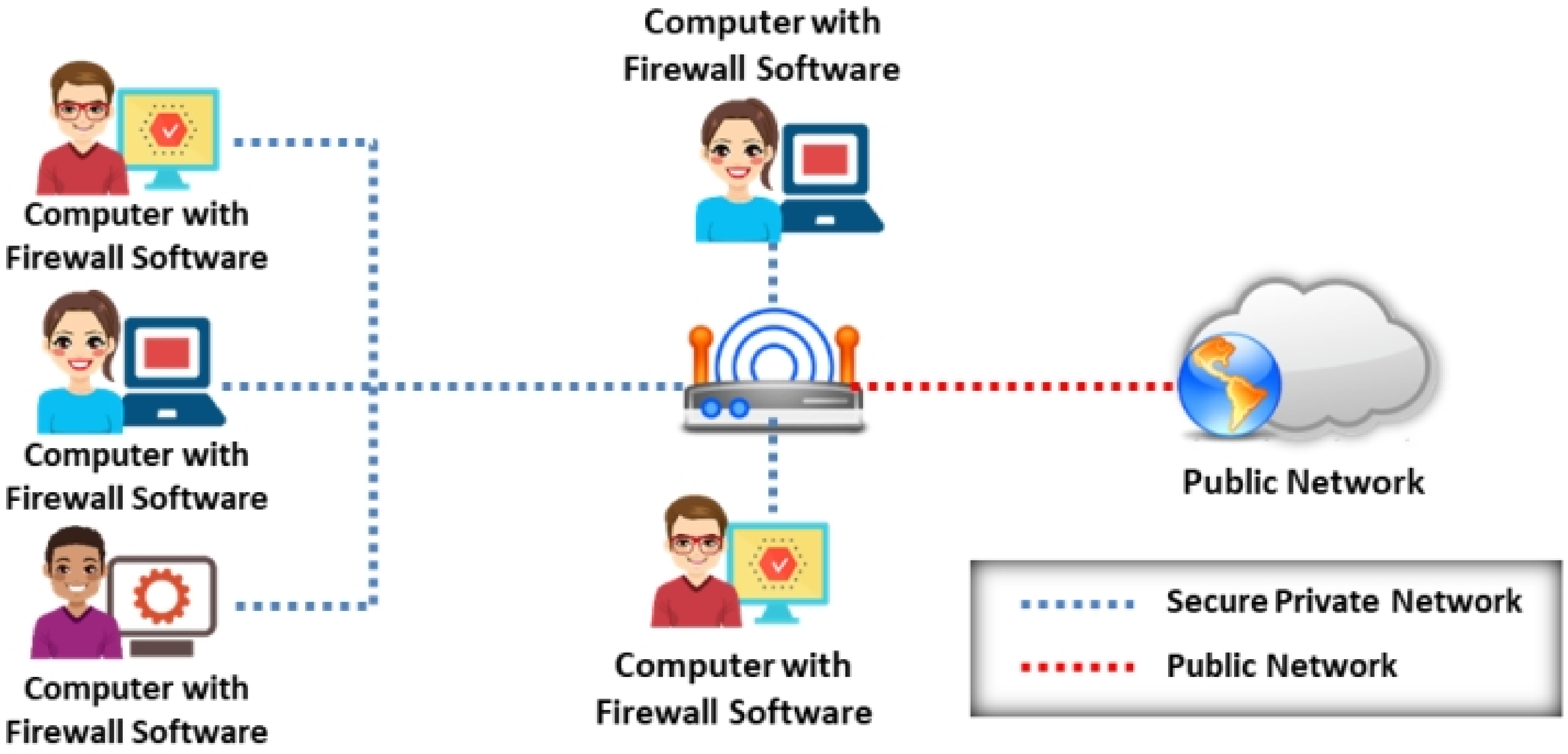
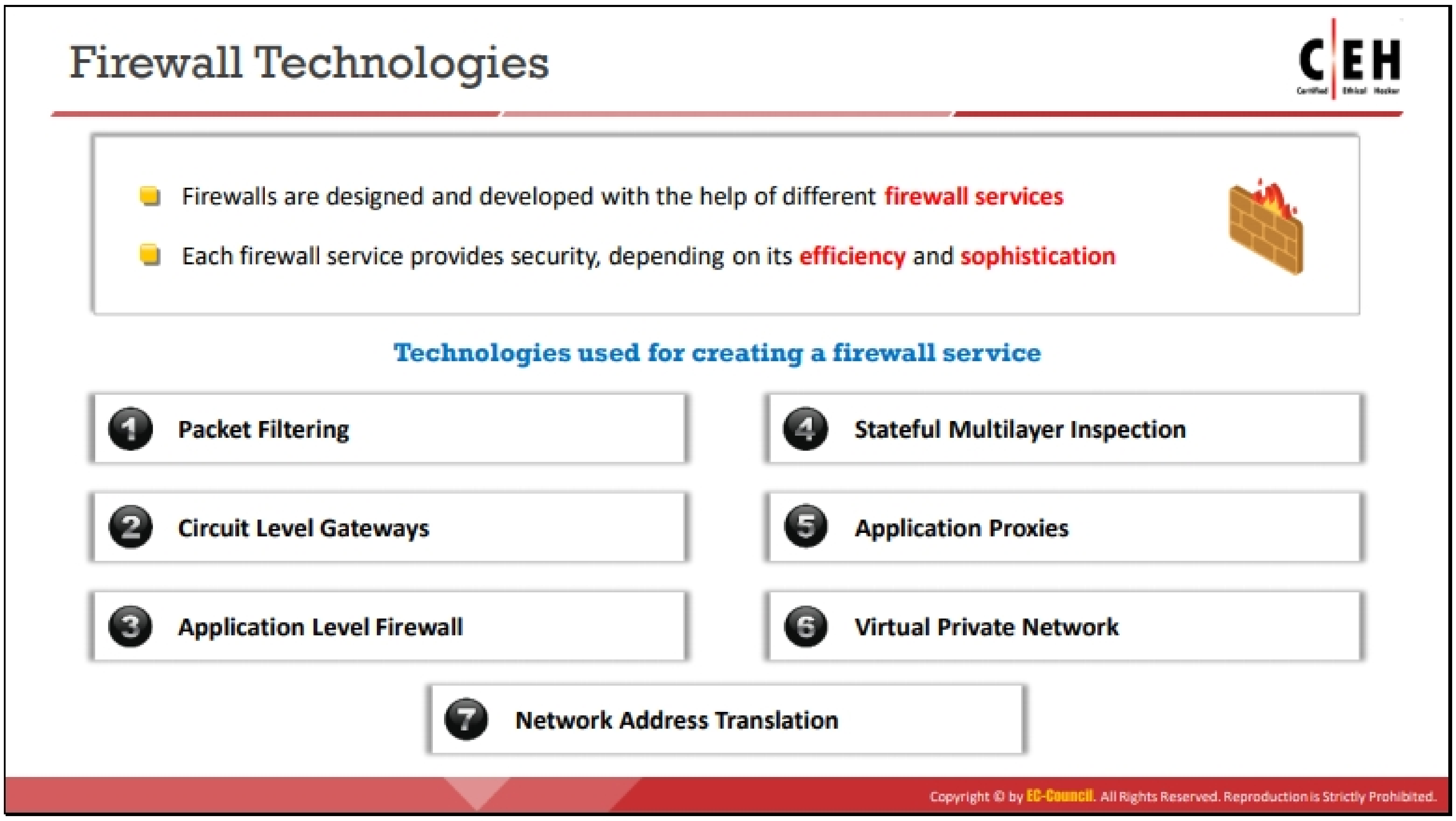

Computer with Firewall Software
Computer with Firewall Software
Computer with Firewall Software
Computer with Firewall Software
Computer with Firewall Software
Public Network
Secure Private Network
Public Network

Figure 12.12: Software Firewall



Firewall Technologies

- Firewalls are designed and developed with the help of different **firewall services**
- Each firewall service provides security, depending on its **efficiency** and **sophistication**

Technologies used for creating a firewall service

1. **Packet Filtering**
2. **Circuit Level Gateways**
3. **Application Level Firewall**
4. **Stateful Multilayer Inspection**
5. **Application Proxies**
6. **Virtual Private Network**
7. **Network Address Translation**

Firewall Technologies

Firewalls are designed and developed with the help of different firewall services. Each firewall service provides security depending on its efficiency and sophistication. There are different types of firewall technologies depending on where the communication is taking place, where the traffic is intercepted in the network, the state that is traced, and so on. Considering the capabilities of different firewalls, it is easy to choose and place an appropriate firewall to meet the security requirements in the best possible way. Each type of firewall has its advantages.

Several firewall technologies are available for organizations to implement their security measures. Sometimes, firewall technologies are combined with other technologies to build another firewall technology. For example, NAT is a routing technology; however, when it is combined with a firewall, it is considered a firewall technology.

The various firewall technologies are listed below:

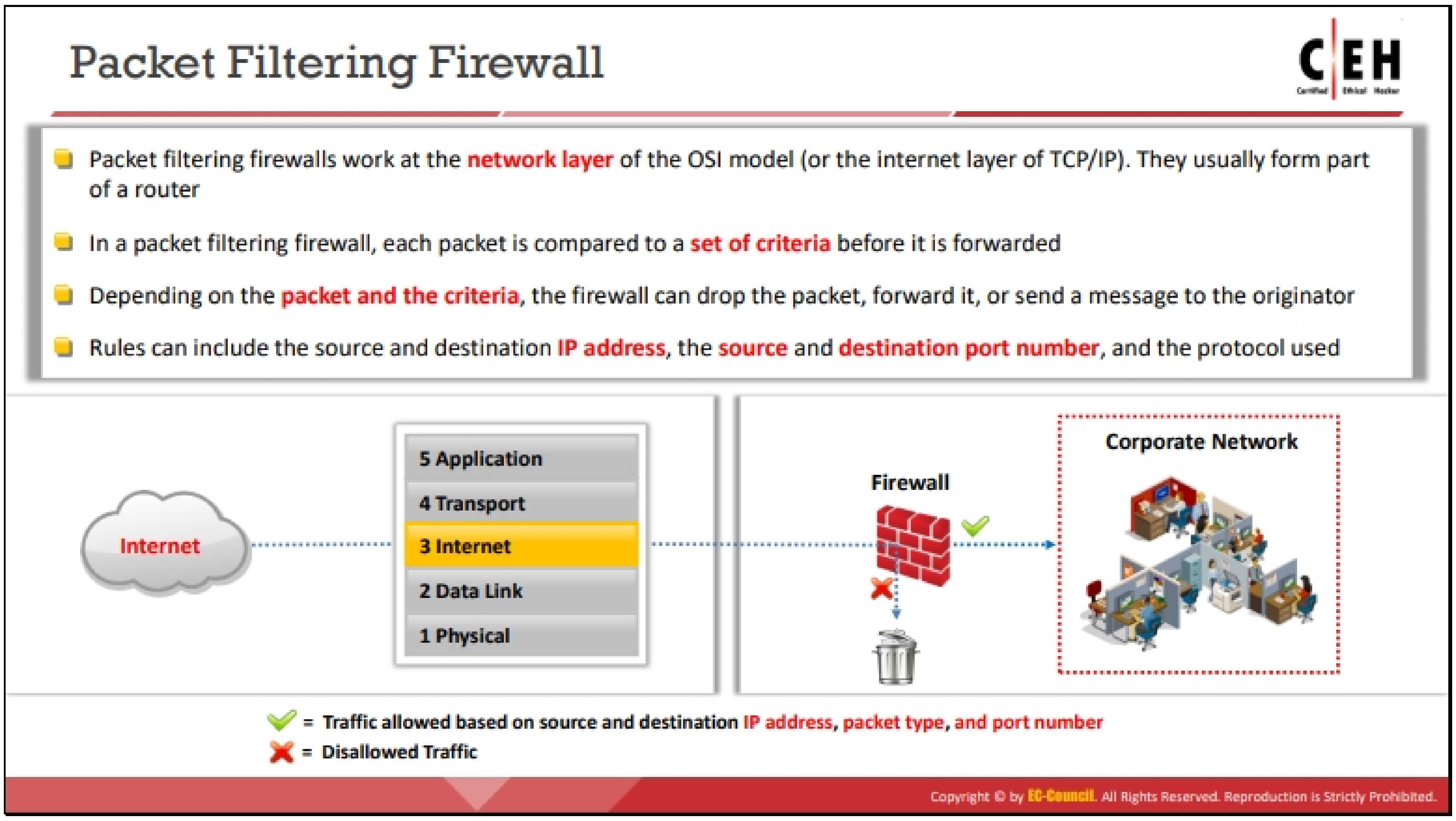
- Packet Filtering
- Circuit-Level Gateways
- Application-Level Firewall
- Stateful Multilayer Inspection
- Application Proxies
- Virtual Private Network
- Network Address Translation

The table below summarizes technologies operating at each OSI layer:

OSI Layer	Firewall Technology
Application	▪ Virtual Private Network (VPN) ▪ Application Proxies
Presentation	▪ Virtual Private Network (VPN)
Session	▪ Virtual Private Network (VPN) ▪ Circuit-Level Gateways
Transport	▪ Virtual Private Network (VPN) ▪ Packet Filtering
Network	▪ Virtual Private Network (VPN) ▪ Network Address Translation (NAT) ▪ Packet Filtering ▪ Stateful Multilayer Inspection
Data Link	▪ Virtual Private Network (VPN) ▪ Packet Filtering
Physical	▪ Not Applicable

Table 12.1: Firewall Technologies

The security levels of these technologies vary according to their efficiency levels. A comparison of these technologies can be made by allowing them to pass through the OSI layer between the hosts. The data passes through the intermediate layers from a higher layer to a lower layer. Each layer adds additional information to the data packets. The lower layer now sends the obtained information through the physical network to the upper layers and then to its destination.



Packet Filtering Firewall

- Packet filtering firewalls work at the **network layer** of the OSI model (or the internet layer of TCP/IP). They usually form part of a router
- In a packet filtering firewall, each packet is compared to a **set of criteria** before it is forwarded
- Depending on the **packet and the criteria**, the firewall can drop the packet, forward it, or send a message to the originator
- Rules can include the source and destination **IP address**, the **source** and **destination port number**, and the protocol used

✓ = Traffic allowed based on source and destination **IP address**, **packet type**, **and port number**
✗ = Disallowed Traffic

Packet Filtering Firewall

In a packet filtering firewall, each packet is compared with a set of criteria before it is forwarded. Depending on the packet and the criteria, the firewall can drop the packet and transmit it or send a message to the originator. The rules can include the source and the destination IP address, the source and the destination port number, and the protocol used. It works at the internet layer of the TCP/IP model or the network layer of the OSI model. Packet filtering firewalls focus on individual packets, analyze their header information, and determine which way they need to be directed. Traditional packet filters make this decision according to the following information in a packet:

- **Source IP address**: Used to check whether the packet is coming from a valid source. The information about the source IP address can found from the IP header of the packet.
- **Destination IP address**: Checks if the packet is going to the correct destination and if the destination accepts these types of packets. The information about the destination IP address can found from the IP header of the packet.
- **Source TCP/UDP port**: Used to check the source port of the packet
- **Destination TCP/UDP port**: Used to monitor the destination port regarding the services to be allowed and the services to be denied.
- **TCP flag bits**: Used to check whether the packet has SYN, ACK, or other bits set for the connection to be made.
- **Protocol in use**: Used to check whether the protocol that the packet is carrying should be allowed.
- **Direction**: Used to check whether the packet is entering or leaving the private network.

- **Interface**: Used to check whether the packet is coming from an unreliable zone.

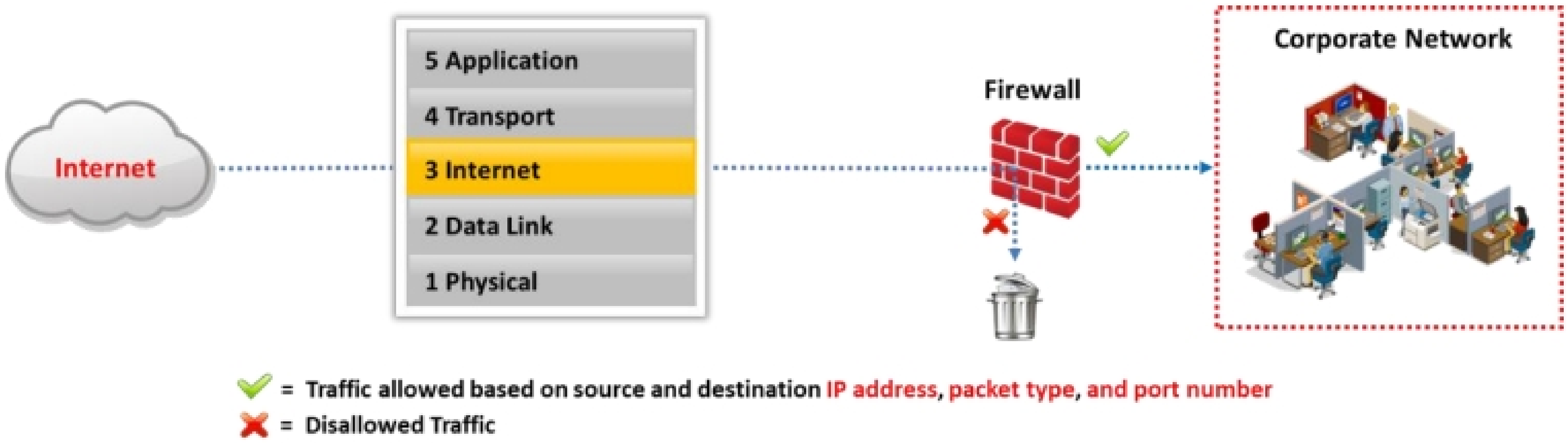
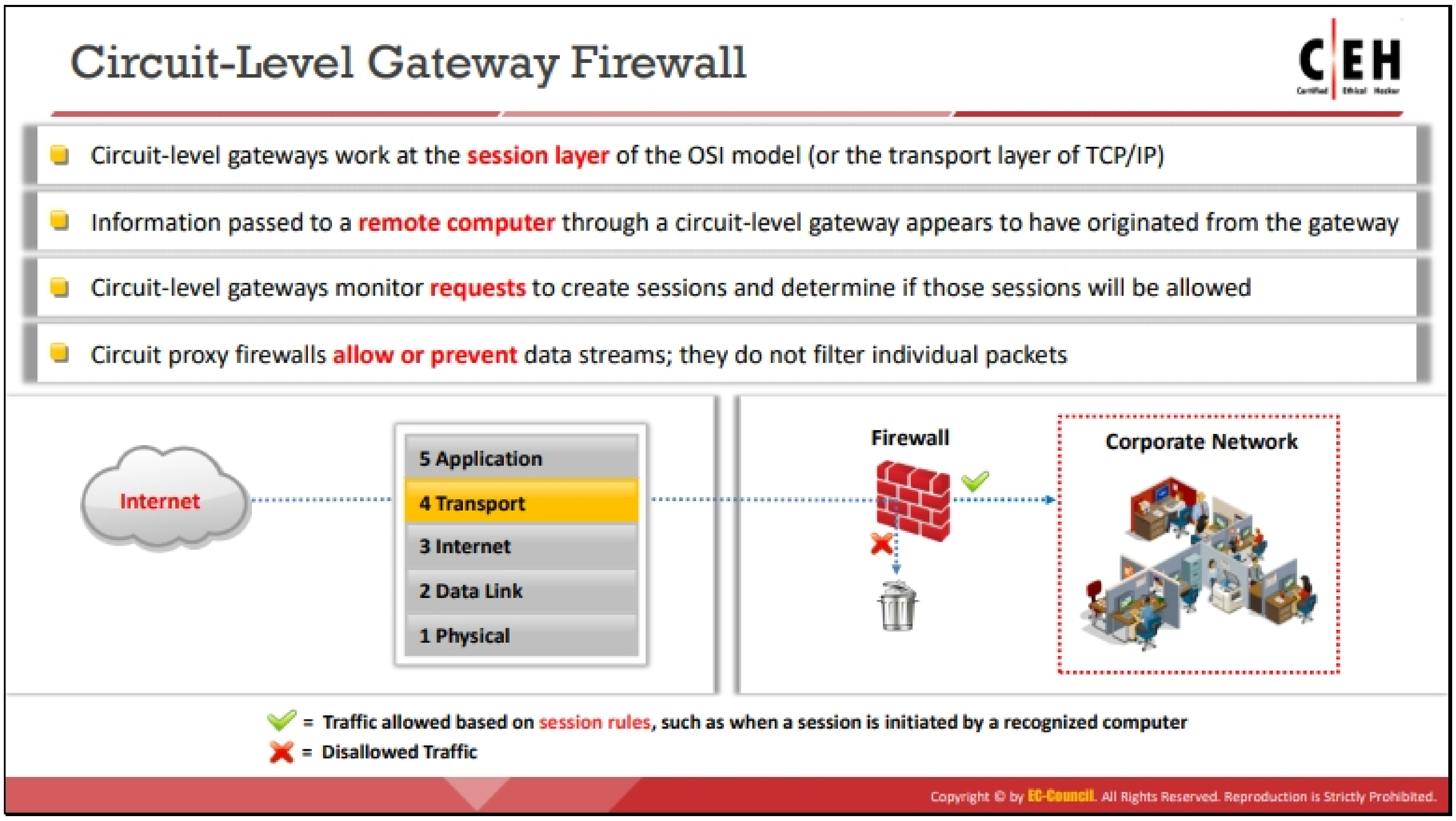

Internet
5 Application
4 Transport
3 Internet
2 Data Link
1 Physical
Firewall
Corporate Network
✓ = Traffic allowed based on source and destination IP address, packet type, and port number
✗ = Disallowed Traffic

Figure 12.13 Example of Packet Filtering Firewall



Circuit-Level Gateway Firewall

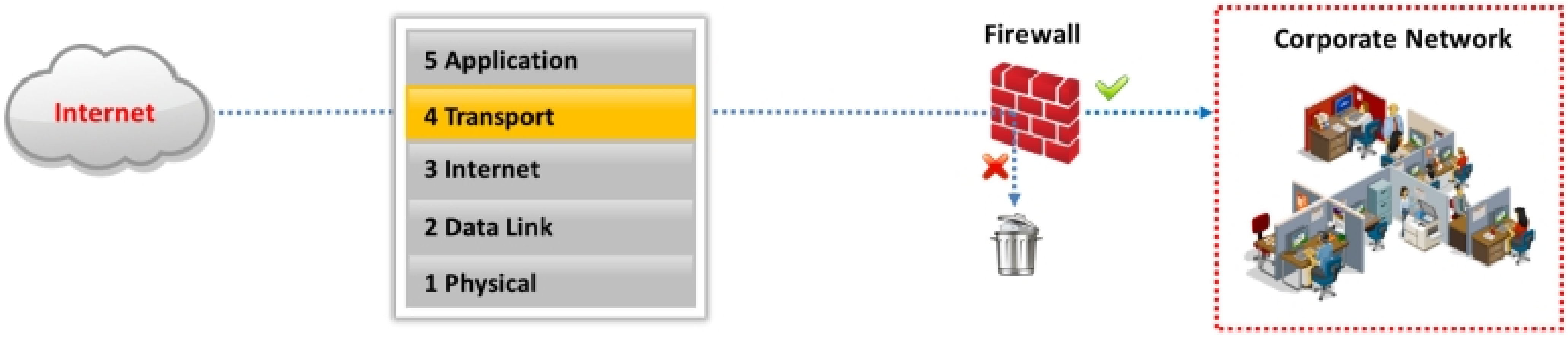
- Circuit-level gateways work at the **session layer** of the OSI model (or the transport layer of TCP/IP)
- Information passed to a **remote computer** through a circuit-level gateway appears to have originated from the gateway
- Circuit-level gateways monitor **requests** to create sessions and determine if those sessions will be allowed
- Circuit proxy firewalls **allow or prevent** data streams; they do not filter individual packets

✓ = Traffic allowed based on session rules, such as when a session is initiated by a recognized computer
✗ = Disallowed Traffic

Circuit-Level Gateway Firewall

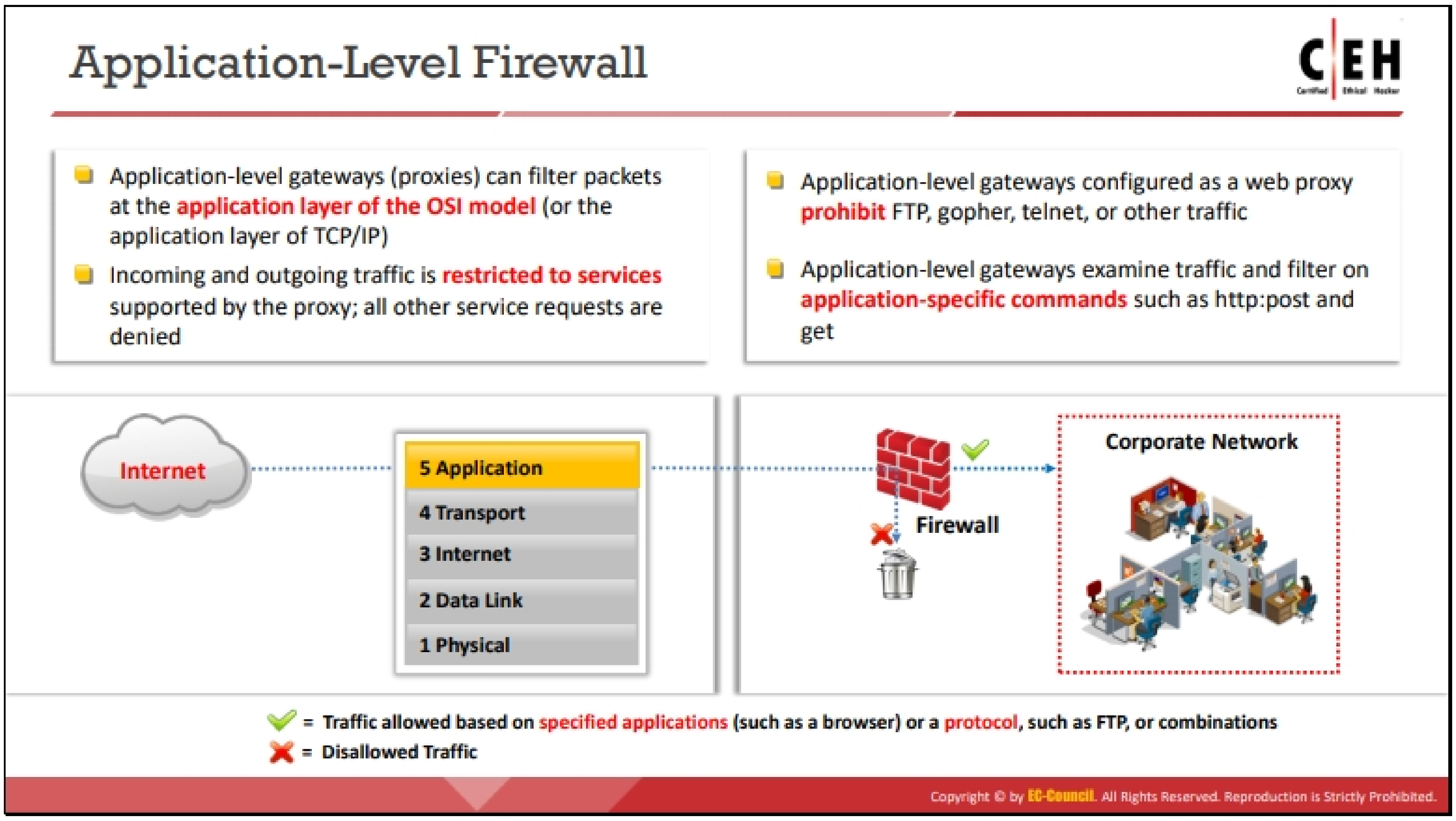
A circuit-level gateway firewall works at the session layer of the OSI model or transport layer of TCP/IP. It forwards data between networks without verification and blocks incoming packets from the host but allows the traffic to pass through itself. Information passed to remote computers through a circuit-level gateway will appear to have originated from the gateway, as the incoming traffic carries the IP address of the proxy (circuit-level gateway). Such firewalls monitor requests to create sessions and determine if those sessions will be allowed.

A circuit-level gateway gives controlled access to network services and host requests. To determine whether a requested session is valid, it checks the TCP handshake between packets. Circuit proxy firewalls allow or prevent data streams; they do not filter individual packets. They are relatively inexpensive and hide the information about the private network that they protect.



✓ = Traffic allowed based on session rules, such as when a session is initiated by a recognized computer
✗ = Disallowed Traffic

Figure 12.14 Example of Circuit-Level Gateway Firewall



Application-Level Firewall

- Application-level gateways (proxies) can filter packets at the **application layer of the OSI model** (or the application layer of TCP/IP)
- Incoming and outgoing traffic is **restricted to services** supported by the proxy; all other service requests are denied
- Application-level gateways configured as a web proxy **prohibit** FTP, gopher, telnet, or other traffic
- Application-level gateways examine traffic and filter on **application-specific commands** such as http:post and get

✓ = Traffic allowed based on specified applications (such as a browser) or a protocol, such as FTP, or combinations
✗ = Disallowed Traffic

Application-Level Firewall

Application-based proxy firewalls focus on the application layer rather than just the packets. Application-level gateways (proxies) can filter packets at the application layer of the OSI model (or the application layer of TCP/IP). Incoming and outgoing traffic is restricted to services supported by the proxy; all other service requests are denied. The need for an application-level firewall arises from the tremendous amount of voice, video, and collaborative traffic in the data-link layer and network layer, which may be used for unauthorized access to internal and external networks. Application-level gateways configured as web proxies prohibit FTP, gopher, telnet, or other traffic. They examine traffic and filter application-specific commands such as HTTP: post and get.

Traditional firewalls are unable to filter such types of traffic. They can inspect, find, and verify malicious traffic that is missed by stateful inspection firewalls to make decisions as to whether to allow access, and they improve the overall security of the application layer. For example, worms that send malicious code in legitimate protocols cannot be detected by stateful firewalls, as proxy firewalls focus on packet headers in the network layer. However, deep packet inspection firewalls can find such attacks with the help of informative signatures added inside packets.

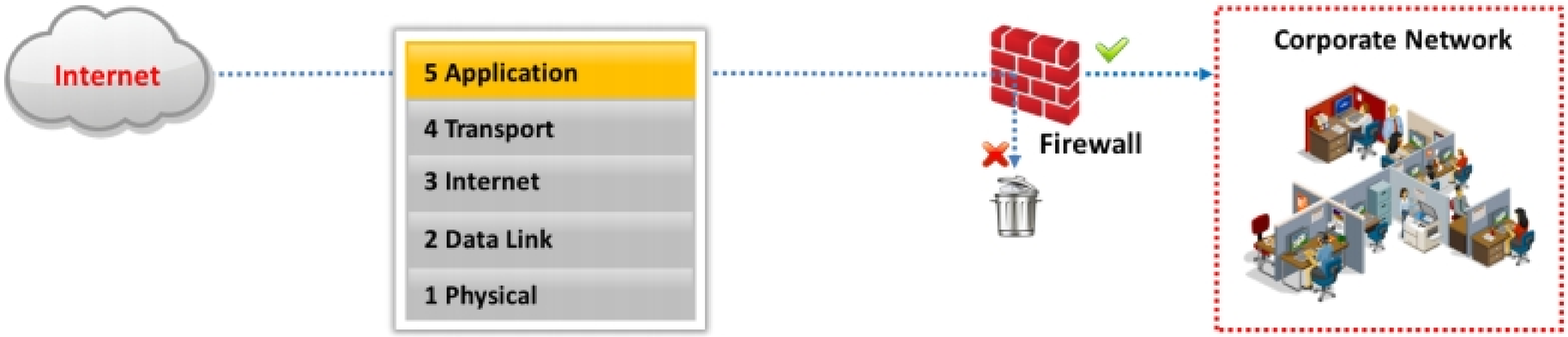
Some of the features of application-level firewalls are as follows:

- They analyze the application information to make decisions as to whether to permit traffic.
- Being proxy-based, they can permit or deny traffic according to the authenticity of the user or process involved.

- A content-caching proxy optimizes performance by caching frequently accessed information rather than sending new requests to the servers for the same old data.

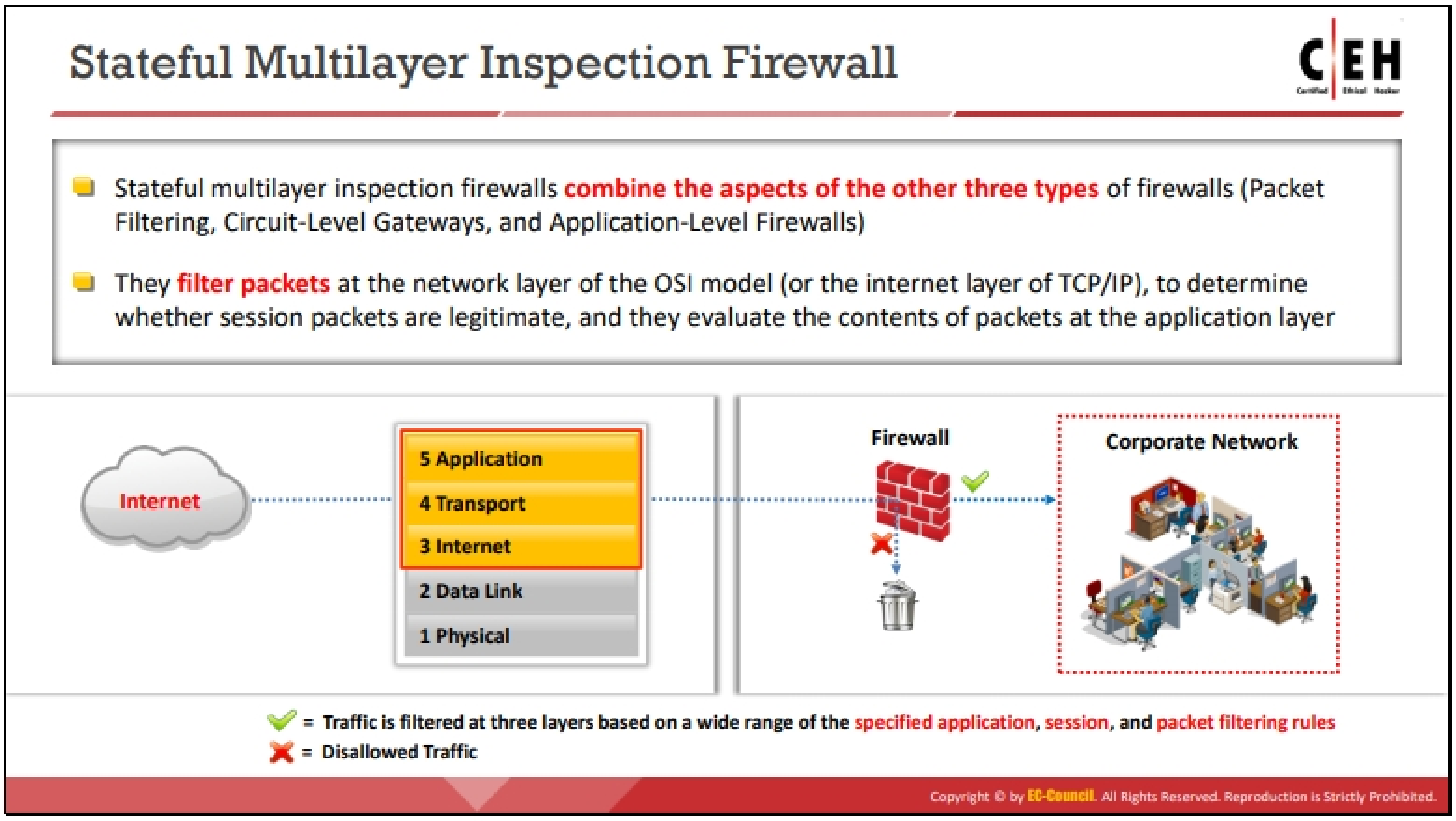
Application-layer firewalls can function in one of two modes: active or passive.

- **Active application-level firewalls**: They examine all incoming requests, including the actual message that is exchanged, against known vulnerabilities, such as SQL injection, parameter and cookie tampering, and cross-site scripting. The requests that are deemed genuine are allowed to pass through them.
- **Passive application-level firewalls**: They work similarly to IDS in that they also check all incoming requests against known vulnerabilities, but they do not actively reject or deny those requests if a potential attack is discovered.



✓ = Traffic allowed based on specified applications (such as a browser) or a protocol, such as FTP, or combinations
✗ = Disallowed Traffic

Figure 12.15: Example of Application-Level Firewall



Stateful Multilayer Inspection Firewall

- Stateful multilayer inspection firewalls **combine the aspects of the other three types** of firewalls (Packet Filtering, Circuit-Level Gateways, and Application-Level Firewalls)
- They **filter packets** at the network layer of the OSI model (or the internet layer of TCP/IP), to determine whether session packets are legitimate, and they evaluate the contents of packets at the application layer

✓ = Traffic is filtered at three layers based on a wide range of the specified application, session, and packet filtering rules
✗ = Disallowed Traffic

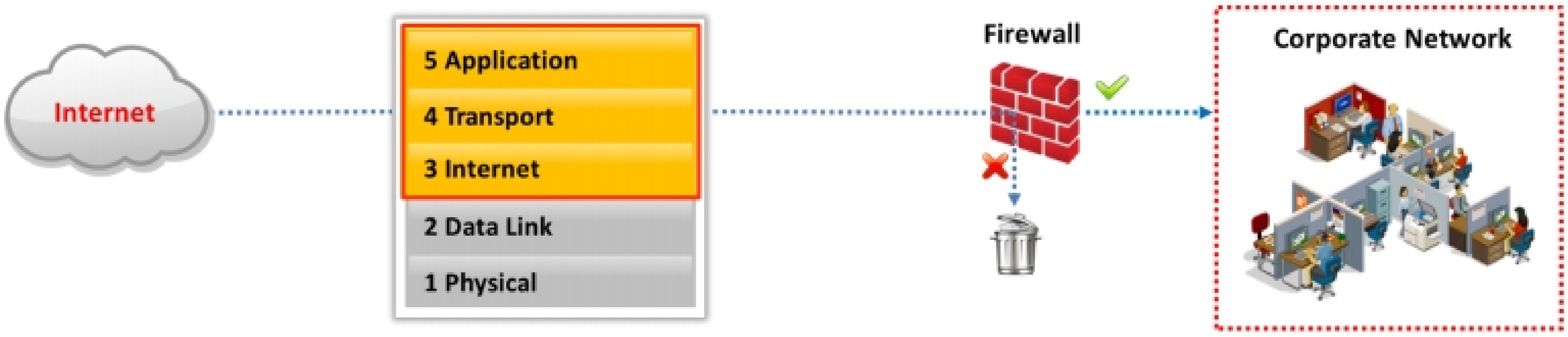
Stateful Multilayer Inspection Firewall

Stateful multilayer inspection firewalls combine the aspects of the three above-mentioned types of firewalls (packet filtering, circuit-level gateways, and application-level firewalls). They filter packets at the network layer of the OSI model (or the internet layer of the TCP/IP model) to determine whether session packets are legitimate, and they evaluate the contents of the packets at the application layer.

Using stateful packet filtering, you can overcome the limitation of packet firewalls, which can only filter the IP address, port, protocol, and so on. This multilayer firewall can perform deep packet inspection.

Features of the Stateful Multilayer Inspection Firewall:

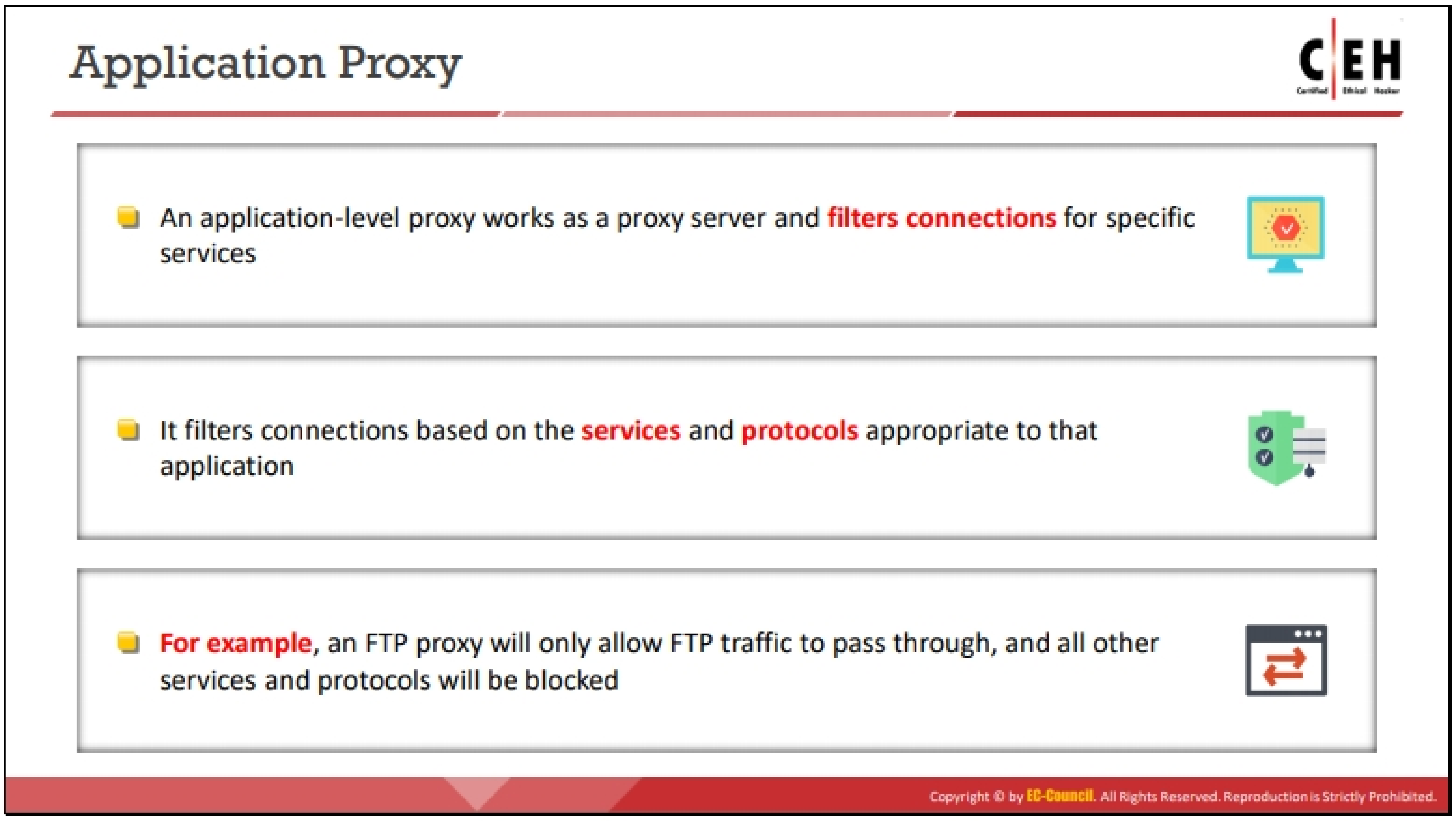
- This type of firewall can remember the packets that passed through it earlier and make decisions about future packets accordingly.
- These firewalls combine the best features of both packet filtering and application-based filtering.
- Cisco PIX firewalls are stateful.
- These firewalls track and log slots or translations.


Internet
5 Application
4 Transport
3 Internet
2 Data Link
1 Physical
Firewall
Corporate Network

✓ = **Traffic is filtered at three layers based on a wide range of the specified application, session, and packet filtering rules**

✗ = **Disallowed Traffic**

Figure 12.16: Example of Stateful Multilayer Inspection Firewall



Application Proxy

- An application-level proxy works as a proxy server and **filters connections** for specific services
- It filters connections based on the **services** and **protocols** appropriate to that application
- **For example**, an FTP proxy will only allow FTP traffic to pass through, and all other services and protocols will be blocked

Application Proxy

An application-level proxy works as a proxy server and filters connections for specific services. It filters connections based on the services and protocols when acting as a proxy. For example, an FTP proxy will only allow FTP traffic to pass through while all other services and protocols will be blocked. It is a type of server that acts as an interface between the user workstation and the Internet. It correlates with the gateway server and separates the enterprise network from the Internet. It receives a request from a user to provide the Internet service and responds to the original request only. A proxy service is an application or program that helps forward user requests (for example, FTP or Telnet) to the actual services. A proxy is also known as an application-level gateway, as it renews the connections and act as a gateway to the services. Proxies run on a firewall host that is either a dual-homed host or some other bastion host for security purposes. Some proxies, namely caching proxies, improve network efficiency. They keep copies of the requested data of the hosts that they proxy. Such proxies can provide the data directly when multiple hosts request the same data. Caching proxies help in reducing the load on network connections, whereas proxy servers provide both security and caching.

A proxy service is available to the user in the internal network and the service in the outside network (Internet), and it is transparent. Instead of direct communication, it talks with the proxy and it handles all the communication between users and Internet services. Transparency is the main advantage of proxy services. To the user, a proxy server presents an illusion that it is dealing directly with the real server, whereas to a real server, the proxy server gives the illusion that it is dealing directly with the user.

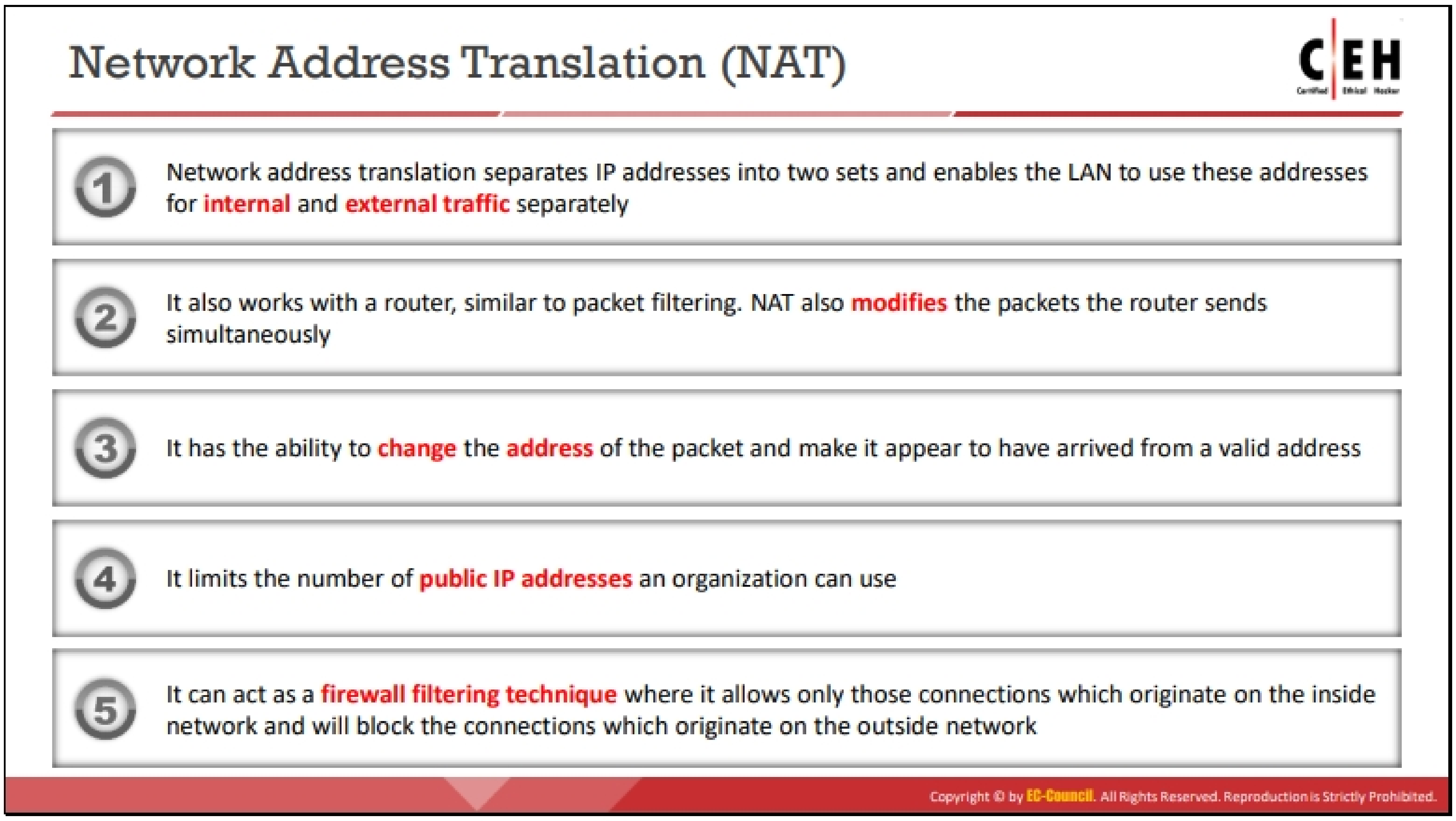
Advantages

- Proxy services are useful for logging because they can understand application protocols and effectively allow logging.

- Proxy services reduce the load on network links as they are capable of caching copies of frequently requested data and allow it to be directly loaded from the system instead of the network.
- Proxy systems perform user-level authentication, as they are involved in the connection.
- Proxy systems automatically protect weak or faulty IP implementations as they sit between the client and the Internet and generate new IP packets for the client.

Disadvantages

- Proxy services lag behind non-proxy services until suitable proxy software is available.
- Each service in a proxy may use different servers.
- Proxy services may require changes in the client, applications, and procedures.



Network Address Translation (NAT)

1. Network address translation separates IP addresses into two sets and enables the LAN to use these addresses for **internal** and **external traffic** separately
2. It also works with a router, similar to packet filtering. NAT also **modifies** the packets the router sends simultaneously
3. It has the ability to **change** the **address** of the packet and make it appear to have arrived from a valid address
4. It limits the number of **public IP addresses** an organization can use
5. It can act as a **firewall filtering technique** where it allows only those connections which originate on the inside network and will block the connections which originate on the outside network

Network Address Translation (NAT)

Network address translation (NAT) separates IP addresses into two sets and enables the LAN to use these addresses for internal and external traffic. The NAT helps hide an internal network layout and force connections to go through a choke point. It also works with a router, and similarly to packet filtering, it will also modify the packets that the router sends simultaneously. When the internal machine forwards the packet to the external machine, the NAT modifies the source address of the packet to make it appear as if it is coming from a valid address. When the external machine sends the packet to the internal machine, the NAT modifies the destination address to turn the visible address into the correct internal address. The NAT can also change the source and destination port numbers. It limits the number of public IP addresses that an organization can use. It can act as a firewall filtering technique whereby it allows only those connections that originate in the internal network and blocks the connections that originate in the external network.

NAT systems use different schemes for translation between internal and external addresses:

- Assign one external host address for each internal address and always apply the same translation. This slows down connections and does not provide any savings in address space.
- Dynamically allocate an external host address without modifying the port numbers when the internal host initiates a connection. This restricts the number of internal hosts that can simultaneously access the Internet to the number of available external addresses.
- Create a fixed mapping from internal addresses to externally visible addresses but use a port mapping so that multiple internal machines use the same external address.

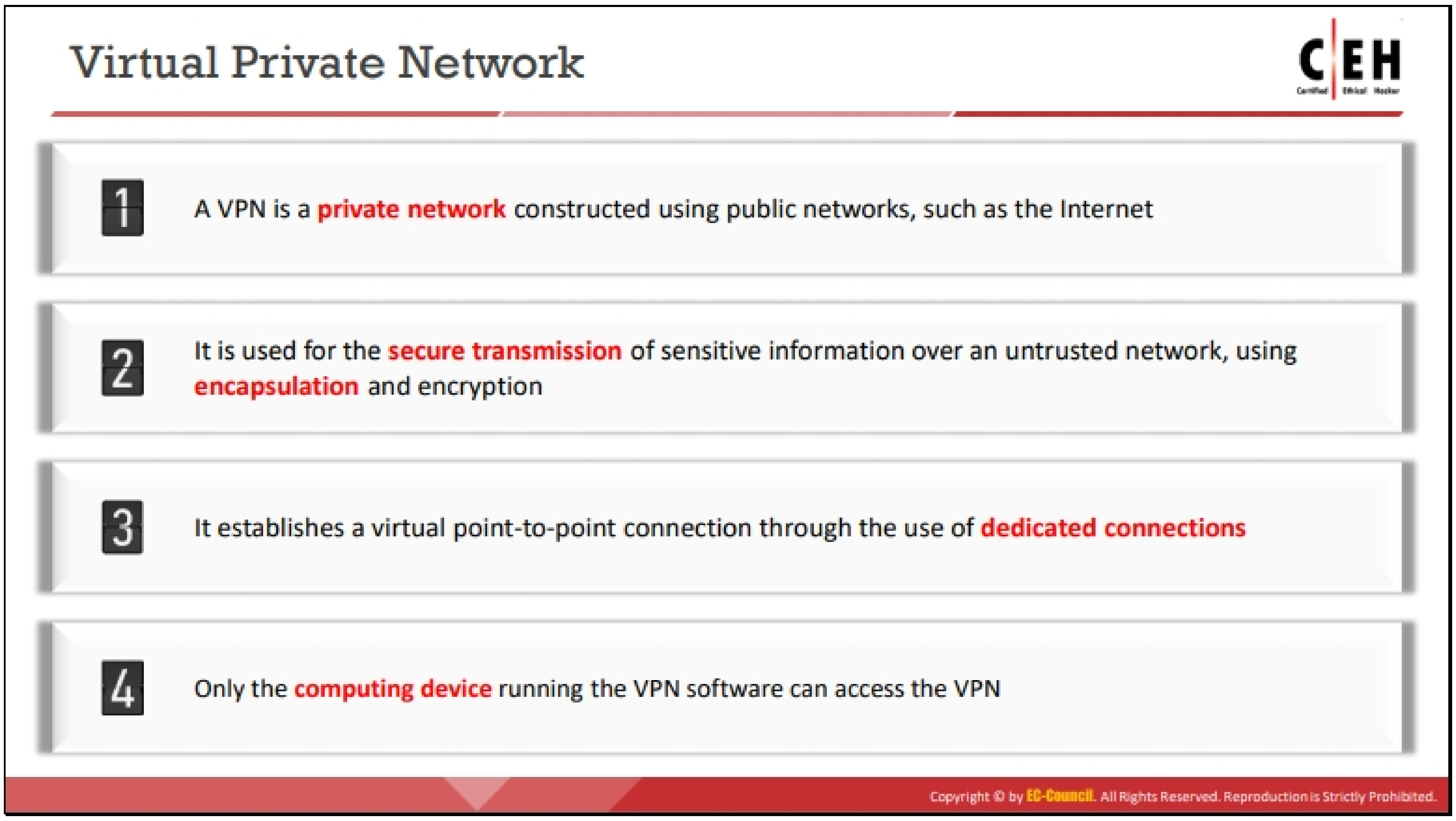
- Dynamically allocate an external host address and port pair each time an internal host initiates a connection. This makes the most efficient possible use of the external host addresses.

Advantages

- Network address translation helps to enforce the firewall's control over outbound connections.
- It restricts incoming traffic and allows only packets that are part of a current interaction initiated from the inside.
- It helps hide the internal network's configuration and thus lowers the success rate of attacks on the network or system.

Disadvantages

- The NAT system has to guess how long it should keep a particular translation, which is not always possible.
- The NAT interferes with encryption and authentication systems to ensure the security of the data.
- Dynamic allocation of ports may interfere with packet filtering.



Virtual Private Network

1. A VPN is a **private network** constructed using public networks, such as the Internet
2. It is used for the **secure transmission** of sensitive information over an untrusted network, using **encapsulation** and encryption
3. It establishes a virtual point-to-point connection through the use of **dedicated connections**
4. Only the **computing device** running the VPN software can access the VPN

Virtual Private Network

A virtual private network (VPN) is a network that provides secure access to the private network through the Internet. VPNs are used for connecting wide area networks (WAN). They allow computers on one network to connect to computers on another network. They are used for the secure transmission of sensitive information over an untrusted network via encapsulation and encryption. They employ encryption and integrity protection, enabling you to use a public network as a private network. A VPN performs encryption and decryption outside the packet-filtering perimeter to allow the inspection of packets coming from other sites. It establishes a virtual point-to-point connection through the use of dedicated connections. A VPN also encapsulates packets sent over the Internet. It combines the advantages of both public and private networks. VPNs have no relation to firewall technology, but firewalls are convenient for adding VPN features as they help in providing secure remote services. The computing device running the VPN software can only access the VPN.

All VPNs that run over the Internet adopt the following principles:

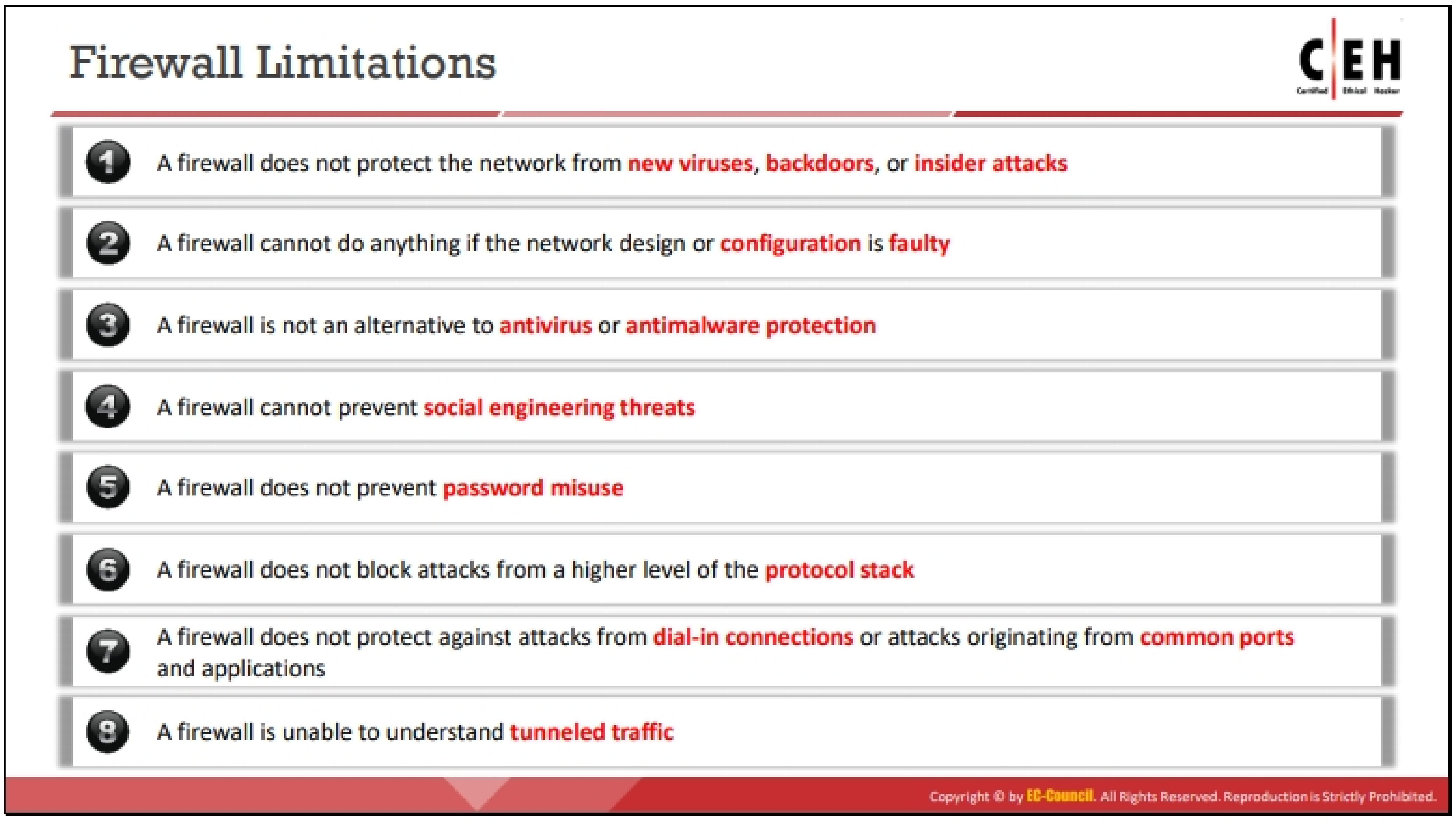
- Encrypts the traffic
- Checks for integrity protection
- Encapsulates new packets, which are sent across the Internet to some destination that reverses the encapsulation
- Checks the integrity
- Decrypts the traffic eventually

Advantages

- A VPN hides all the traffic that flows over it, ensures encryption, and protects data from snooping.
- It provides remote access for protocols while avoiding attackers from the Internet at large.

Disadvantages

- As the VPN runs on a public network, the user will be vulnerable to an attack on the destination network.



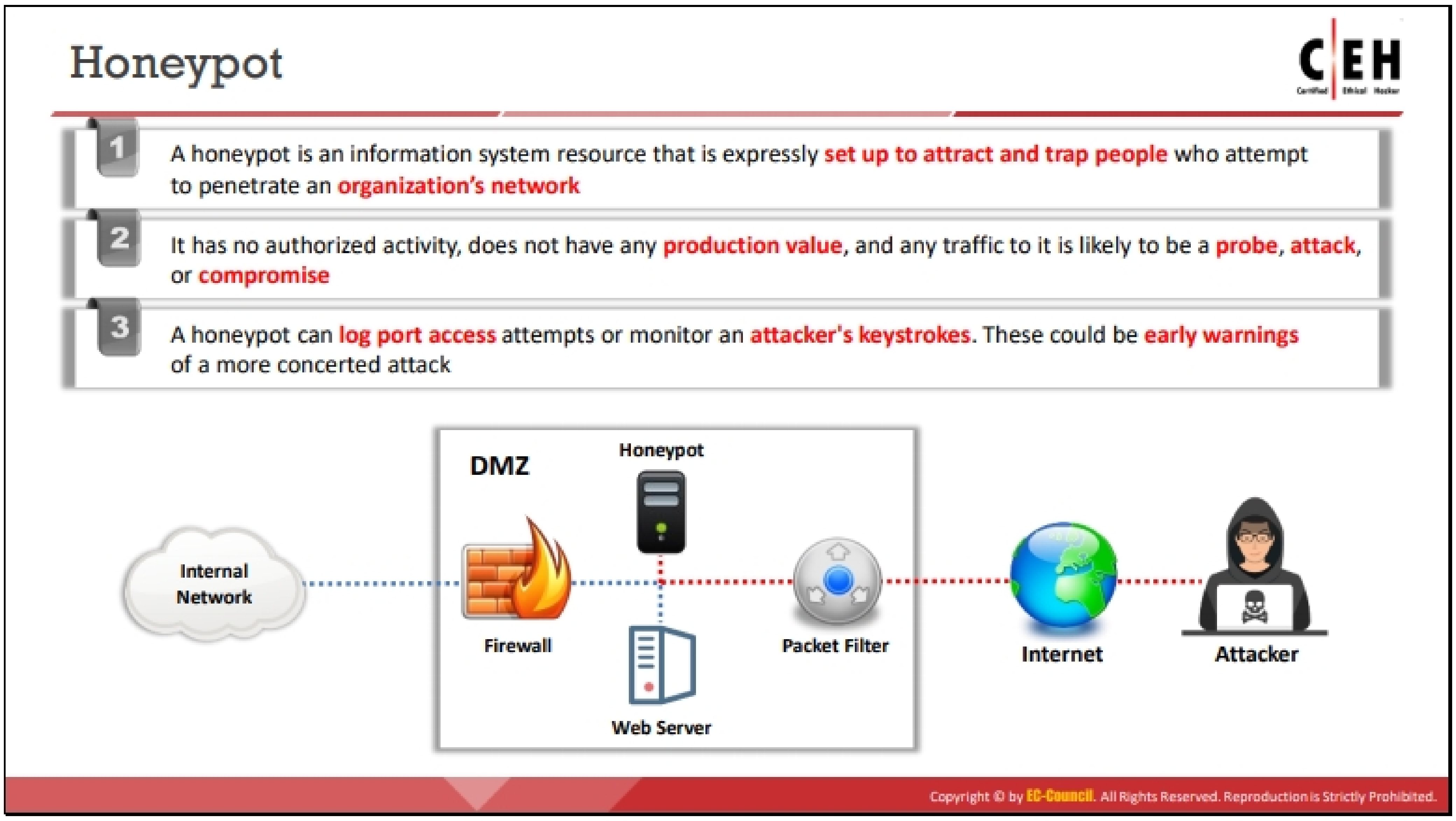
Firewall Limitations

1. A firewall does not protect the network from **new viruses**, **backdoors**, or **insider attacks**
2. A firewall cannot do anything if the network design or **configuration** is **faulty**
3. A firewall is not an alternative to **antivirus** or **antimalware protection**
4. A firewall cannot prevent **social engineering threats**
5. A firewall does not prevent **password misuse**
6. A firewall does not block attacks from a higher level of the **protocol stack**
7. A firewall does not protect against attacks from **dial-in connections** or attacks originating from **common ports** and applications
8. A firewall is unable to understand **tunneled traffic**

Firewall Limitations

Although firewalls are essential to your security strategy, they have the following limitations:

- Firewalls can restrict users from accessing valuable services such as FTP, Telnet, NIS, etc., and they sometimes restrict Internet access as well.
- The firewall cannot prevent internal attacks (backdoor) in a network, e.g., a disgruntled employee who cooperates with the external attacker.
- The firewall focuses its security at a single point, which makes other systems within the network prone to security attacks.
- A bottleneck could occur if all the connections pass through the firewall.
- The firewall cannot protect the network from social engineering and data-driven attacks whereby the attacker sends malicious links and emails to employees inside the network.
- If external devices such as laptops, mobile phones, portable hard drives, etc., are already infected and connected to the network, then a firewall cannot protect the network from these devices.
- The firewall is unable to adequately protect the network from all types of zero-day viruses that try to bypass it.
- A firewall cannot do anything if the network design and configuration is faulty.
- A firewall is not an alternative to antivirus or antimalware tools.
- A firewall does not block attacks from a higher level of the protocol stack.
- A firewall does not prevent attacks originating from common ports and applications.
- A firewall does not prevent attacks from dial-in connections.
- A firewall is unable to understand tunneled traffic.



Honeypot

1. A honeypot is an information system resource that is expressly **set up to attract and trap people** who attempt to penetrate an **organization’s network**
2. It has no authorized activity, does not have any **production value**, and any traffic to it is likely to be a **probe**, **attack**, or **compromise**
3. A honeypot can **log port access** attempts or monitor an **attacker's keystrokes**. These could be **early warnings** of a more concerted attack

Honeypot

A honeypot is a computer system on the Internet intended to attract and trap those who attempt unauthorized or illicit utilization of the host system to penetrate an organization’s network. It is a fake proxy run to frame attackers by logging traffic through it and then sending complaints to the victims’ ISPs. It has no authorized activity or production value, and any traffic to it is likely a probe, attack, or compromise. Whenever there is any interaction with a honeypot, it is most likely to be malicious. Honeypots are unique; they do not solve a specific problem. Instead, they are a highly flexible tools with many different security applications. Honeypots help in preventing attacks, detecting attacks, and information gathering and research. A honeypot can log port access attempts or monitor an attacker's keystrokes; these could be early warnings of a more concerted attack. It requires a considerable amount of effort to maintain a honeypot.

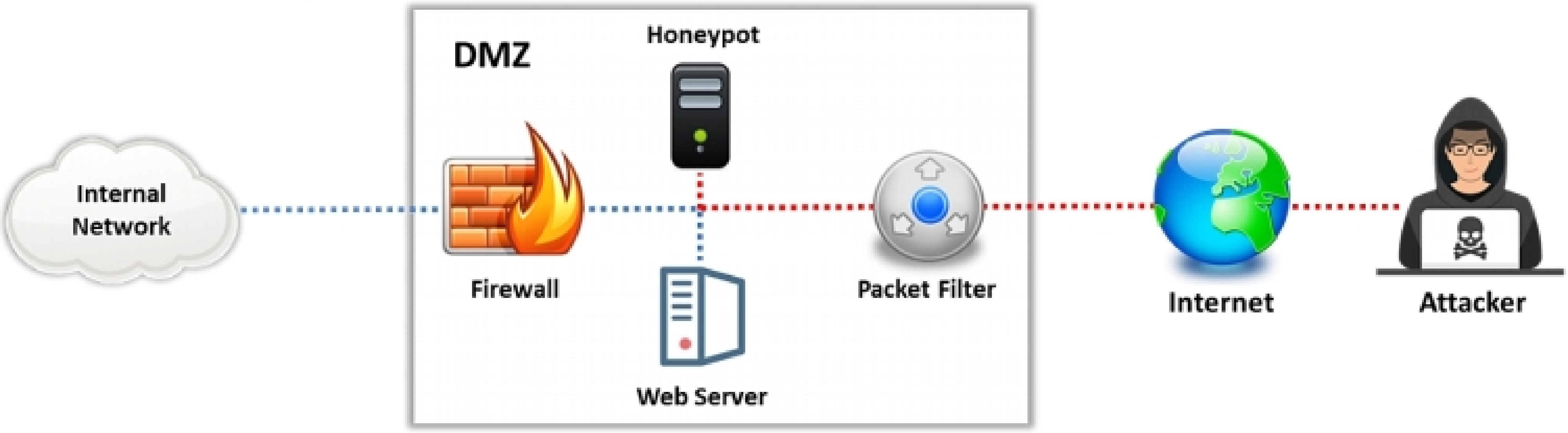
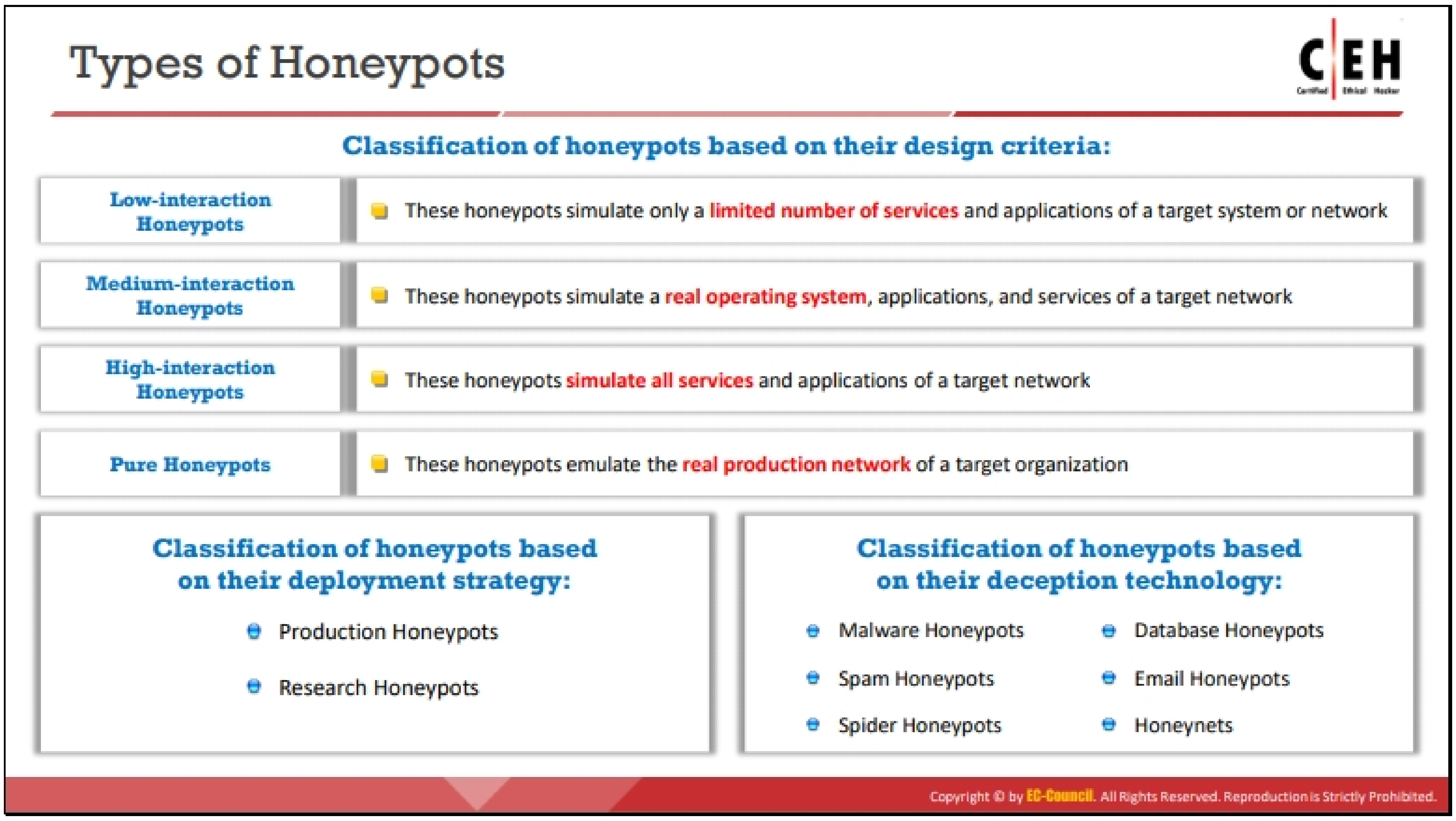


Figure 12.17: Example of Honeypot



Types of Honeypots

Classification of honeypots based on their design criteria:

Low-interaction Honeypots	These honeypots simulate only a **limited number of services** and applications of a target system or network
Medium-interaction Honeypots	These honeypots simulate a **real operating system**, applications, and services of a target network
High-interaction Honeypots	These honeypots **simulate all services** and applications of a target network
Pure Honeypots	These honeypots emulate the **real production network** of a target organization

Classification of honeypots based on their deployment strategy:

- Production Honeypots
- Research Honeypots

Classification of honeypots based on their deception technology:

- Malware Honeypots
- Spam Honeypots
- Spider Honeypots
- Database Honeypots
- Email Honeypots
- Honeynets

Types of Honeypots

Honeypots are classified into the following types based on their design criteria:

- **Low-interaction Honeypots**

  Low-interaction honeypots emulate only a limited number of services and applications of a target system or network. If the attacker does something that the emulation does not expect, the honeypot will simply generate an error. They capture limited amounts of information, i.e., mainly transactional data, and some limited interactions. These honeypots cannot be compromised completely. They are set to collect higher-level information about attack vectors such as network probes and worm activities. Some examples are Specter, KFSensor, and Honeytrap.

  KFSensor is a low-interaction honeypot used to attract and identify penetrations. It implements vulnerable system services and Trojans to attract hackers. This honeypot can be used to monitor all TCP, UDP, and ICMP ports and services. KFSensor identifies and raises alerts about port scanning and DoS attacks.

  A honeytrap is a low-interaction honeypot used to observe attacks against TCP and UDP services. It runs as a daemon and starts server processes dynamically on requested ports. Attackers are tricked into sending responses to the honeytrap server process. The data that is received by the honeypot is concatenated into a string and stored in a database file. This string is called the attack string. Honeytraps parse attack strings for a command requesting the server to download a file from another host in the network. If such a command is detected, the server tries to access the corresponding file automatically. It supports only FTP and TFTP protocols. It also identifies and logs HTTP_URIs.

- **Medium-interaction Honeypots**

  Medium-interaction honeypots simulate a real OS as well as applications and services of a target network. They provide greater misconception of an OS than low-interaction honeypots. Therefore, it is possible to log and analyze more complex attacks. These honeypots capture more useful data than low-interaction honeypots. They can only respond to preconfigured commands; therefore, the risk of intrusion increases. The main disadvantage of medium-interaction honeypots is that the attacker can quickly discover that the system behavior is abnormal. Some examples of medium-interaction honeypots include HoneyPy, Kojoney2, and Cowrie.

  Kojoney2 is a medium-interaction honeypot that emulates a real SSH environment. This honeypot listens on port 21 for incoming SSH connections. If a connection request is initiated, Kojoney2 will verify users against an internal list of fake users. Usually, the connections are accepted by granting access to the SSH shell. It simulates many shell commands to trick attackers. Using Kojoney2, attackers can download files using wget and curl commands.

- **High-Interaction Honeypots**

  Unlike their low- and medium-interaction counterparts, high-interaction honeypots do not emulate anything; they run actual vulnerable services or software on production systems with real OS and applications. These honeypots simulate all services and applications of a target network. They can be completely compromised by attackers to gain full access to the system in a controlled area. They capture complete information about an attack vector such as attack techniques, tools, and intent. The honeypotized system is more prone to infection, as attack attempts can be carried out on real production systems.

  A honeynet is a prime example of a high-interaction honeypot. It is neither a product nor a software solution that a user installs. Instead, it is an architecture—an entire network of computers designed to attack. The idea is to have an architecture that creates a highly controlled network with real computers running real applications, in which all activities are monitored and logged.

  "**Bad guys**" find, attack, and break into these systems through their own initiative. When they do, they do not realize that they are in a honeynet. Without the knowledge of the attackers, all their activities and actions, from encrypted SSH sessions to email and file uploads, are captured by inserting kernel modules into their systems.

  At the same time, the honeynet controls the attacker's activity. Honeynets do this by using a honeywall gateway, which allows inbound traffic to the victim's systems but controls the outbound traffic using intrusion prevention technologies. This gives the attacker the flexibility to interact with the victim's systems but prevents the attacker from harming other non-honeynet computers.

- **Pure Honeypots**

  Pure honeypots emulate the real production network of a target organization. They cause attackers to devote their time and resources toward attacking the critical production system of the company. Attackers uncover and discover the vulnerabilities and trigger alerts that help network administrators to provide early warnings of attacks and hence reduce the risk of an intrusion.

Honeypots are classified into the following types based on their deployment strategy:

- **Production Honeypots**

  Production honeypots are deployed inside the production network of the organization along with other production servers. Although such honeypots improve the overall state of security of the organization, they effectively capture only a limited amount of information related to the adversaries. Such honeypots fall under the low-interaction honeypot category and are extensively employed by large organizations and corporations. As production honeypots are deployed internally, they also help to find out internal flaws and attackers within an organization.

- **Research Honeypots**

  Research honeypots are high-interaction honeypots primarily deployed by research institutes, governments, or military organizations to gain detailed knowledge about the actions of intruders. By using such honeypots, security analysts can obtain in-depth information about how an attack is performed, vulnerabilities are exploited, and attack techniques and methods are used by the attackers. This analysis, in turn, can help an organization to improve attack prevention, detection, and security mechanisms and develop a more secure network infrastructure.

  The main drawback of research honeypots is that they do not contribute to the direct security of the company. If a company is looking to improve its production infrastructure, it should opt for production honeypots.

Honeypots are classified into the following types based on their deception technology:

- **Malware Honeypots**

  Malware honeypots are used to trap malware campaigns or malware attempts over the network infrastructure. These honeypots are simulated with known vulnerabilities such as outdated APIs, vulnerable SMBv1 protocols, etc., and they also emulate different Trojans, viruses, and backdoors that encourage adversaries to perform exploitation activities. These honeypots lure the attacker or malware into performing attacks, from which the attack pattern, malware signatures, and malware threat actors can be identified effectively.

- **Database Honeypots**

  Database honeypots employ fake databases that are vulnerable to perform database-related attacks such as SQL injection and database enumeration. These fake databases trick the attackers by making them think that these databases contain crucial sensitive information such as credit card details of all the customers and employee databases. However, all the information present in the database are fake and simulated. Such databases lure the attacker to perform attacks, with their vulnerabilities; from the attacks, the attack pattern and the threat actor's TTP's towards database attacks can be identified effectively.

- **Spam Honeypots**

  Spam honeypots specifically target spammers who abuse vulnerable resources such as open mail relays and open proxies. Basically, spam honeypots consist of mail servers that deliberately accept emails from any random source from the Internet. They provide crucial information about spammers and their activities.

- **Email Honeypots**

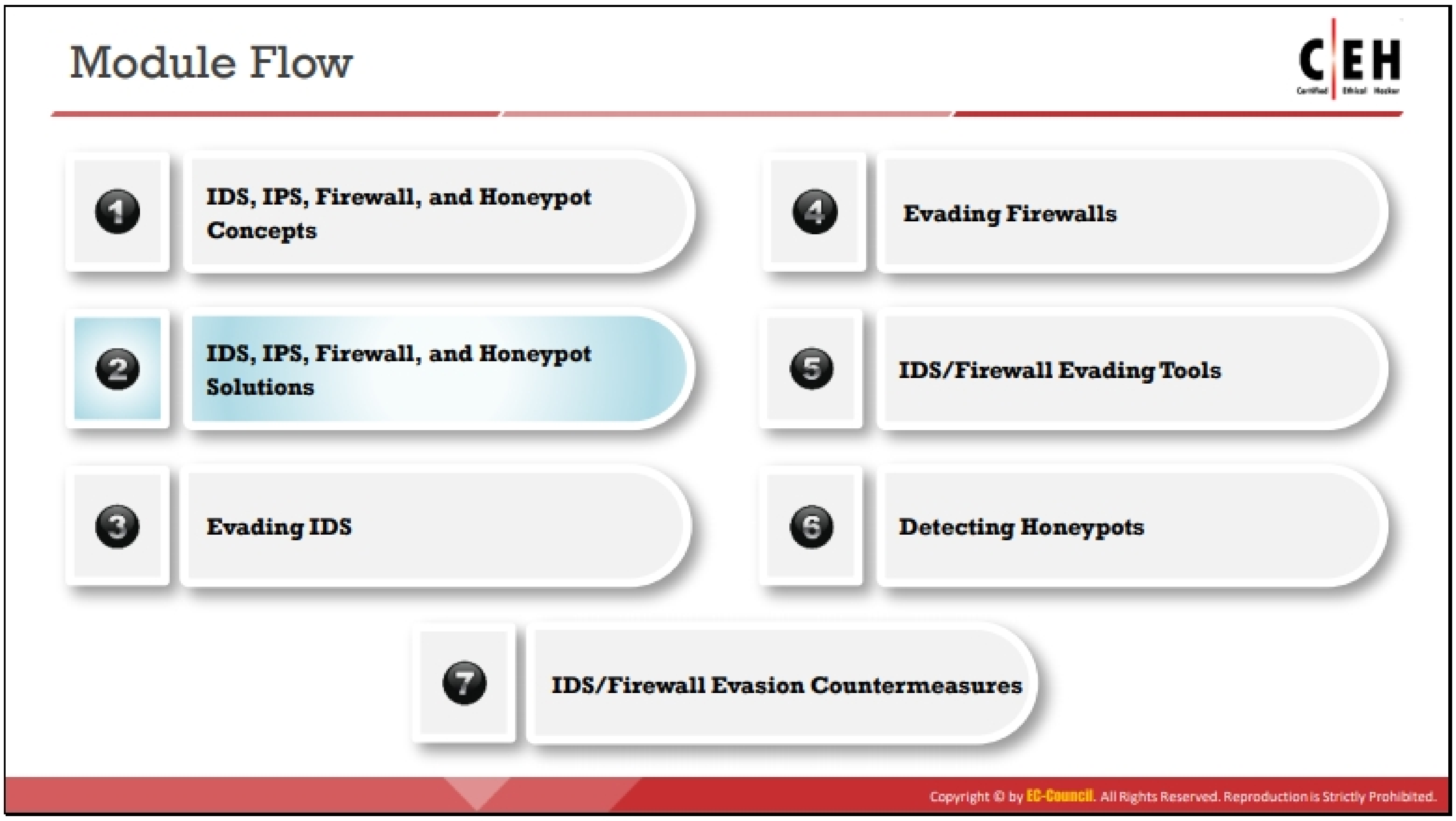
  Email honeypots are also called email traps. They are nothing but fake email addresses that are specifically used to attract fake and malicious emails from adversaries. These fake email IDs will be distributed across the open Internet and dark web to lure threat actors into performing various malicious activities to exploit the organization. By constantly monitoring the incoming emails, the adversary's deception techniques can be identified by the administrators and internal employees can be warned to avoid falling into such email traps.

- **Spider Honeypots**

  Spider honeypots are also called spider traps. These honeypots are specifically designed to trap web crawlers and spiders. Many threat actors perform web crawling and spidering to extract important information from web applications. Such crucial information includes URLs, contact details, directory details, etc. Spider honeypots are employed to trap such adversaries. A fake website will be emulated and presented as a legitimate one. Threat actors attempting to perform web crawling on such traps will be identified and blacklisted.

- **Honeynets**

  Honeynets are networks of honeypots. They are very effective in determining the entire capabilities of the adversaries. Honeynets are mostly deployed in an isolated virtual environment along with a combination of vulnerable servers. The various TTPs employed by different attackers to enumerate and exploit networks will be recorded, and this information can be very effective in determining the complete capabilities of the adversary.



Module Flow

1. IDS, IPS, Firewall, and Honeypot Concepts
2. IDS, IPS, Firewall, and Honeypot Solutions
3. Evading IDS
4. Evading Firewalls
5. IDS/Firewall Evading Tools
6. Detecting Honeypots
7. IDS/Firewall Evasion Countermeasures

IDS, IPS, Firewall, and Honeypot Solutions

The previous section discussed the function, role, and placement of IDS, IPS, firewalls, and honeypots for securing networks. A number of easy-to-use and feature-enriched solutions (hardware, software, or both) are available for the implementation of IDS, IPS, firewalls, and honeypots. This section discusses some commercially available solutions that simplify the usage of IDS, IPS, firewalls, and honeypots.

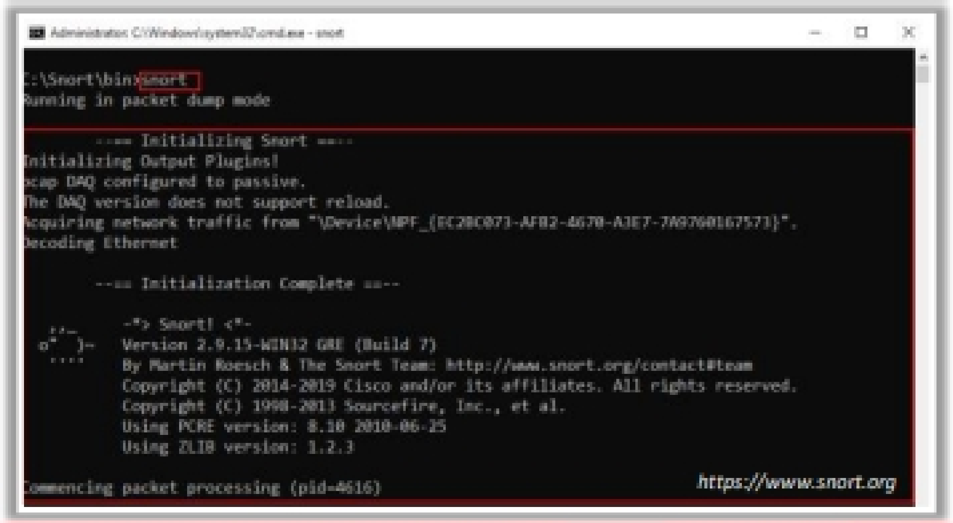
Intrusion Detection Tools: Snort

1. Snort is an open-source network intrusion detection system, capable of performing real-time **traffic analysis and packet logging on IP networks**

2. It can perform **protocol analysis** and **content searching/matching**, and is used to detect a variety of **attacks and probes**, such as buffer overflows, stealth port scans, and OS fingerprinting attempts

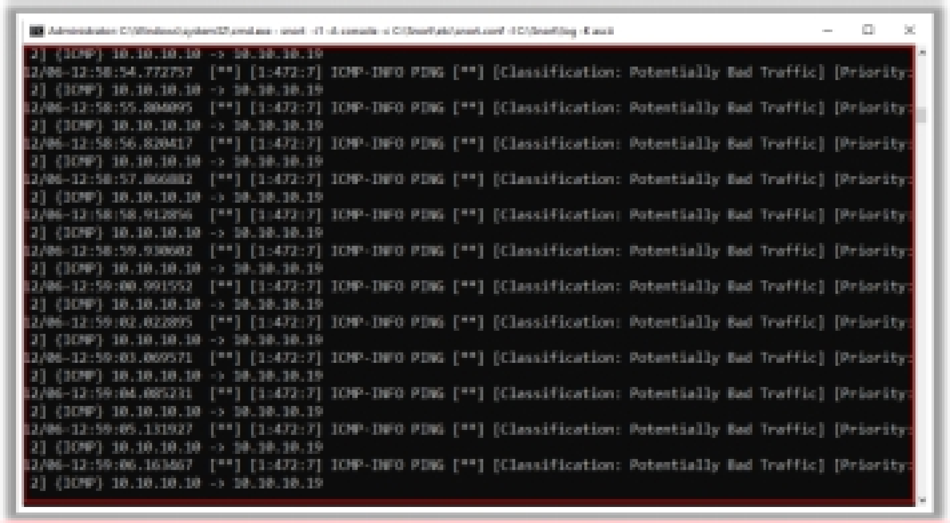
3. It uses a flexible **rules language** to describe traffic that it should collect or pass, as well as a **detection engine** that utilizes a modular plug-in architecture

4. **Uses of Snort**:
   - Straight packet sniffer like tcpdump
   - Packet logger (useful for network traffic debugging, etc.)
   - Network intrusion prevention system



```
:\Snort\bin>snort
unning in packet dump mode

        --== Initializing Snort ==--
nitializing Output Plugins!
cap DAQ configured to passive.
he DAQ version does not support reload.
cquiring network traffic from "\Device\NPF_{EC28C073-AFB2-4670-A3E7-7A9760167573}".
ecoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.15-WIN32 GRE (Build 7)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

ommencing packet processing (pid=4616)
```

https://www.snort.org

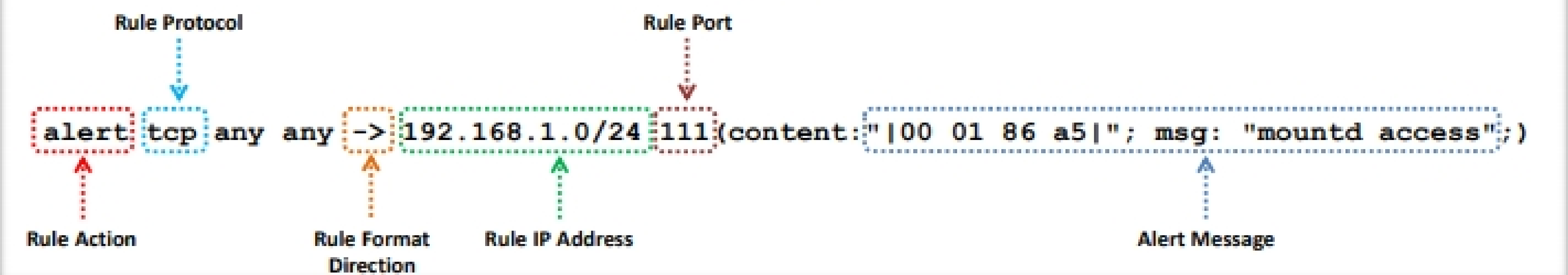


```
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:54.772757  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:55.806095  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:56.820417  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:57.866882  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:58.912856  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:59.930602  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:00.991552  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:02.022895  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:03.069571  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:04.085231  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:05.131927  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:06.163467  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
2] {ICMP} 10.10.10.10 -> 10.10.10.19
```

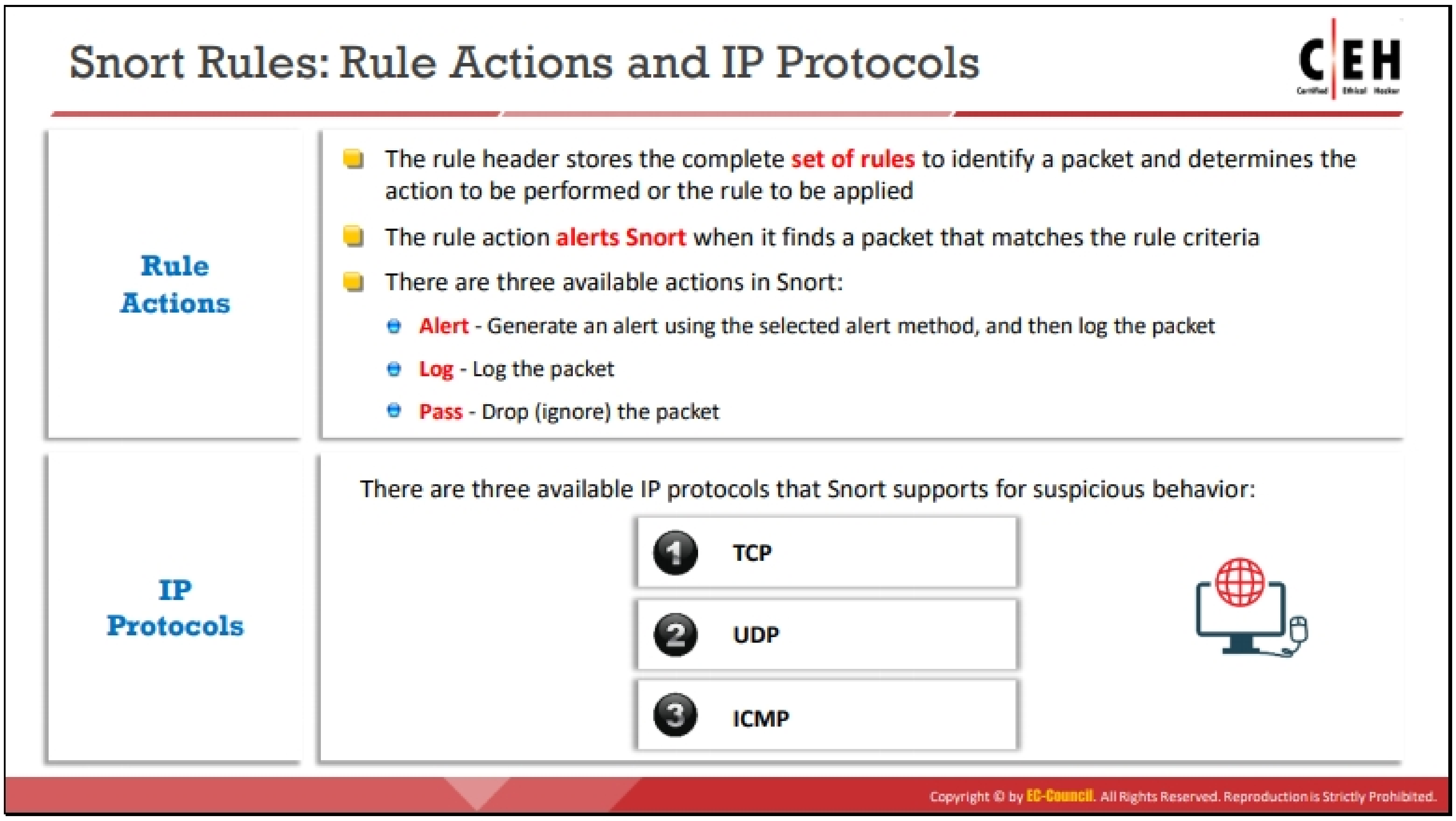
Snort Rules

- Snort's rule engine allows **custom rules** to be established to meet the needs of the network
- Snort rules help in differentiating between **normal Internet activities** and **malicious activities**
- Snort rules must be contained on a **single line**; the Snort rule parser **does not handle rules on multiple lines**
- Snort rules come with two logical parts:
  - **Rule header**: Identifies the **rule's actions**, such as alert, log, pass, activate, and dynamic
  - **Rule options**: Identifies the rule's **alert messages**

Example



```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg: "mountd access";)
```



Snort Rules: Rule Actions and IP Protocols

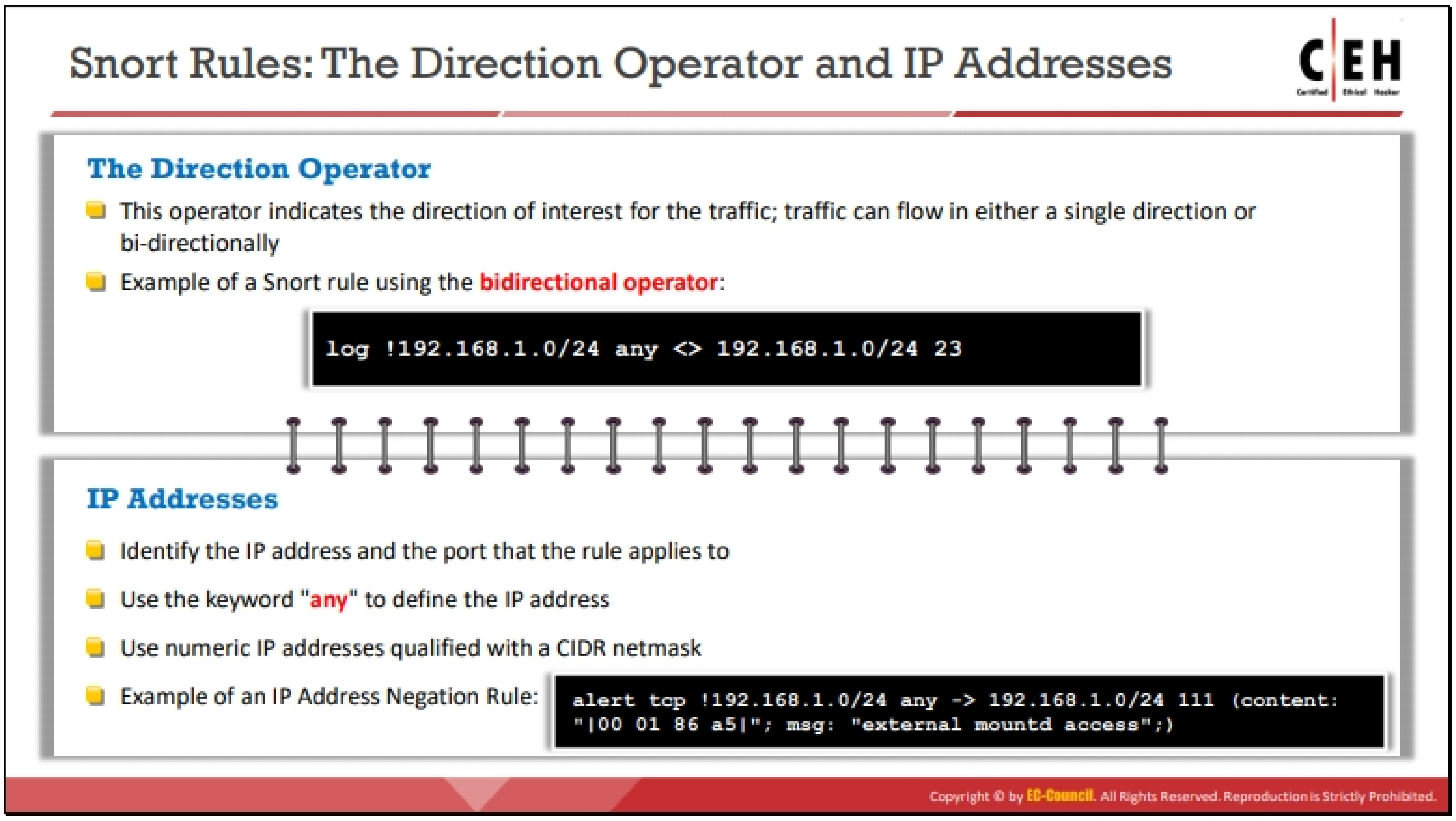
Rule Actions

- The rule header stores the complete **set of rules** to identify a packet and determines the action to be performed or the rule to be applied
- The rule action **alerts Snort** when it finds a packet that matches the rule criteria
- There are three available actions in Snort:
  - **Alert** - Generate an alert using the selected alert method, and then log the packet
  - **Log** - Log the packet
  - **Pass** - Drop (ignore) the packet

IP Protocols

There are three available IP protocols that Snort supports for suspicious behavior:

1. TCP
2. UDP
3. ICMP



Snort Rules: The Direction Operator and IP Addresses

The Direction Operator

- This operator indicates the direction of interest for the traffic; traffic can flow in either a single direction or bi-directionally
- Example of a Snort rule using the **bidirectional operator**:

```
log !192.168.1.0/24 any <> 192.168.1.0/24 23
```

IP Addresses

- Identify the IP address and the port that the rule applies to
- Use the keyword "**any**" to define the IP address
- Use numeric IP addresses qualified with a CIDR netmask
- Example of an IP Address Negation Rule:

```
alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 111 (content:
"|00 01 86 a5|"; msg: "external mountd access";)
```

Snort Rules: Port Numbers

- Port numbers can be listed in different ways, including "**any**" ports, static port definitions, port ranges, and by negation
- Port ranges are indicated with the **range operator ":"**
- Example of a Port Negation

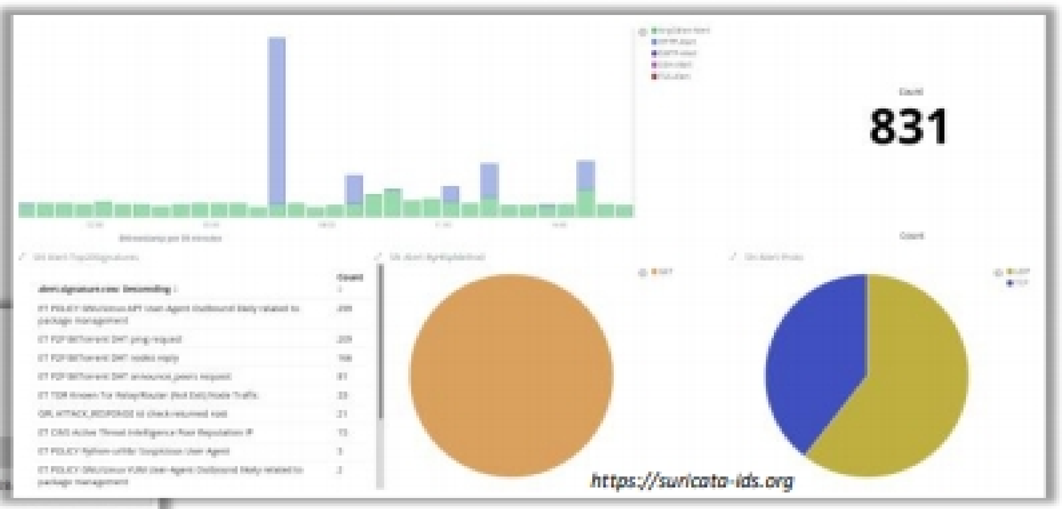
```
log tcp any any -> 192.168.1.0/24 !6000:6010
```

Protocols	IP Address	Action
Log UDP any any ->	192.168.1.0/24 1:1024	Log UDP traffic coming from any port and destination ports ranging from 1 to 1024
Log TCP any any ->	192.168.1.0/24 :5000	Log TCP traffic from any port going to ports less than or equal to 5000
Log TCP any :1024 ->	192.168.1.0/24 400:	Log TCP traffic from the well-known ports and going to ports greater than or equal to 400

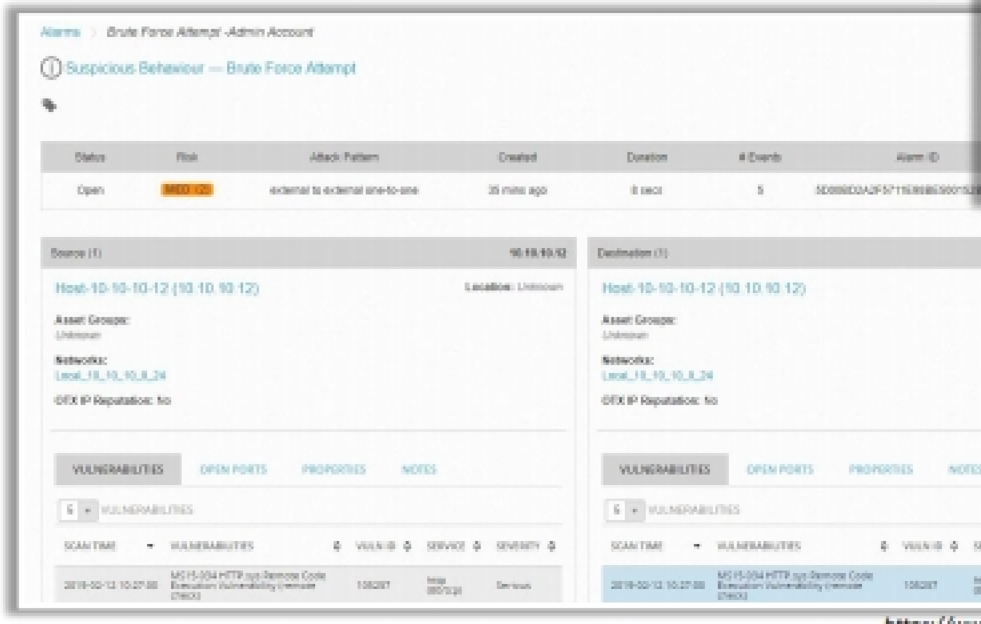
Intrusion Detection Tools: Suricata and AlienVault® OSSIM™

Suricata

- Suricata is a robust network threat detection engine capable of **real-time intrusion detection (IDS)**, **inline intrusion prevention (IPS)**, **network security monitoring (NSM)**, and **offline pcap processing**



https://suricata-ids.org



https://www.alienvault.com

AlienVault® OSSIM™

- AlienVault® OSSIM™ provides you with a feature-rich open source **security information and event management (SIEM) system** complete with event collection, **normalization**, and **correlation**

Intrusion Detection Tools

- SolarWinds Security Event Manager (*https://www.solarwinds.com*)
- OSSEC (*https://www.ossec.net*)
- BroIDS/Zeek IDS (*https://www.zeek.org*)

Intrusion Detection Tools

Intrusion detection tools detect anomalies. These tools, when running on a dedicated workstation, read all network packets, reconstruct user sessions, and scan for possible intrusions by looking for attack signatures and network traffic statistical anomalies. Moreover, these tools offer real-time, zero-day protection from network attacks and malicious traffic, and they prevent malware, spyware, port scans, viruses, DoS, and DDoS from compromising hosts.

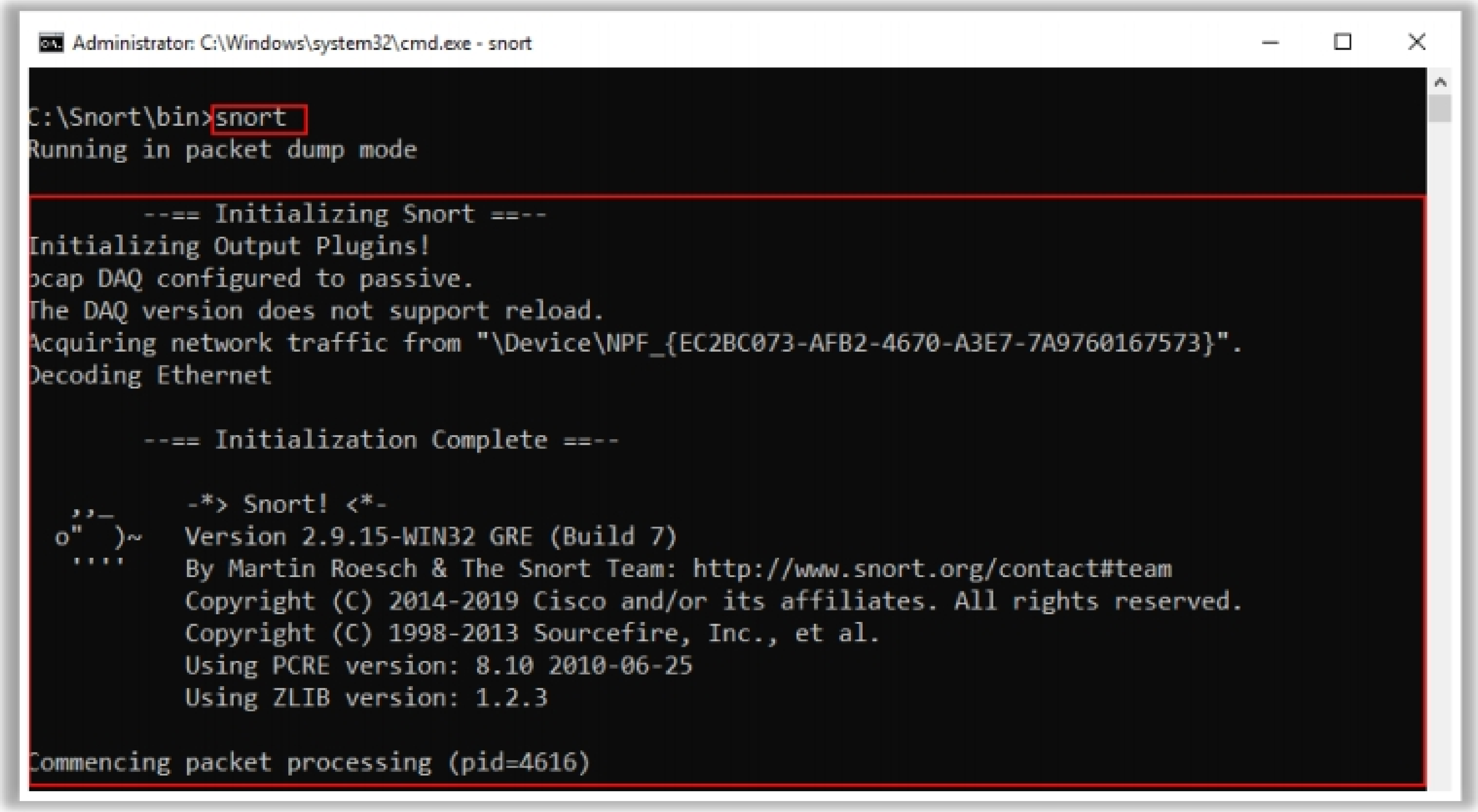
- **Snort**

  Source: *https://www.snort.org*

  Snort is an open-source network intrusion detection system capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis and content searching/matching, and it is used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and OS fingerprinting attempts. It uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that uses a modular plug-in architecture.

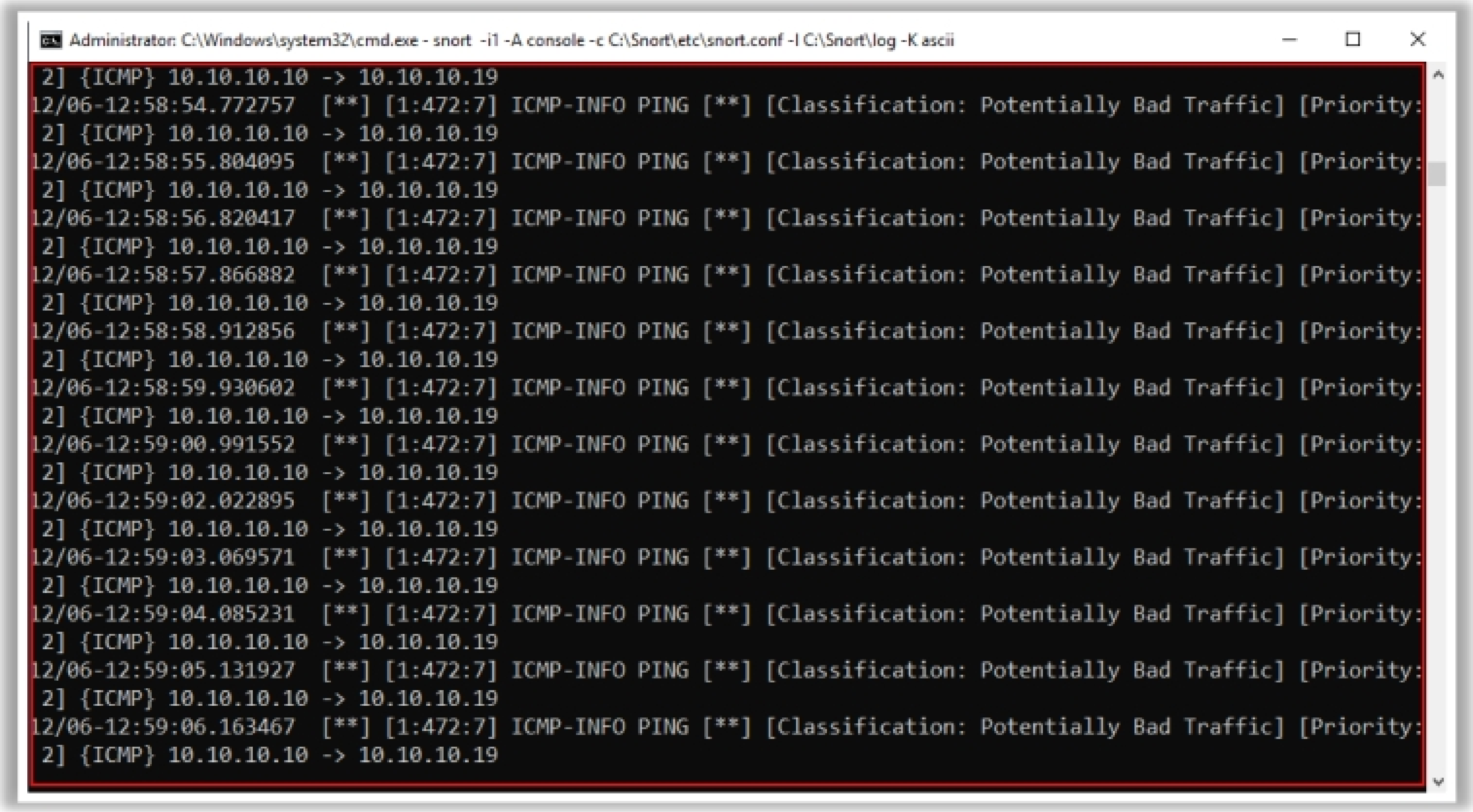
  Uses of Snort:

  - Straight packet sniffer such as tcpdump
  - Packet logger (useful for network traffic debugging, etc.)
  - Network intrusion prevention system



```
C:\Snort\bin>snort
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{EC2BC073-AFB2-4670-A3E7-7A9760167573}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.15-WIN32 GRE (Build 7)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

Commencing packet processing (pid=4616)
```

Figure 12.18: Screenshot of Snort



Administrator: C:\Windows\system32\cmd.exe - snort -i1 -A console -c C:\Snort\etc\snort.conf -l C:\Snort\log -K ascii

```
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:54.772757  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:55.804095  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:56.820417  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:57.866882  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:58.912856  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:58:59.930602  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:00.991552  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:02.022895  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:03.069571  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:04.085231  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:05.131927  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
12/06-12:59:06.163467  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority:
 2] {ICMP} 10.10.10.10 -> 10.10.10.19
```

Figure 12.19: Snort output

Snort Rules

Snort's rule engine allows custom rules to meet the needs of the network. Snort rules help in differentiating between normal Internet activities and malicious activities. Snort uses the popular **libpcap library** (for UNIX/Linux) or **Winpcap** (for Windows), the same library that tcpdump uses to perform its packet sniffing. Attaching Snort in the promiscuous mode to the network media decodes all the packets passing through the network. It generates alerts according to the content of individual packets and rules defined in the configuration file.

Snort allows users to write their own rules. However, each of these Snort rules must describe the following:

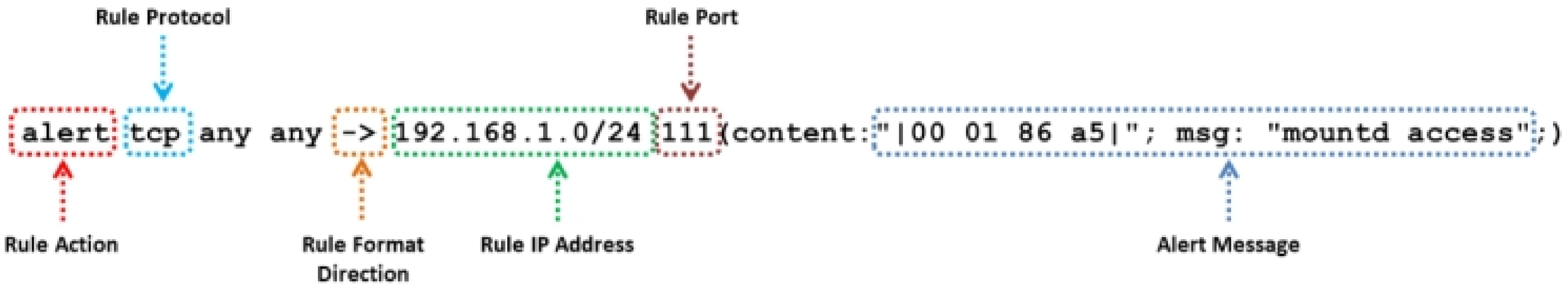
- Any violation of the security policy of the company that might be a threat to the security of the company's network and other valuable information
- All well-known and frequent attempts to exploit the vulnerabilities in the company's network
- The conditions in which a user thinks that a network packet(s) is unusual (i.e., if the identity of the packet is not authentic)

Snort rules, written for both protocol analysis and content searching and matching, should be robust and flexible. The rules should be "**robust**": the system should maintain a hard check on the activities taking place on the network and notify the administrator of any potential intrusion attempt. The rules should be "**flexible**": the system must be sufficiently compatible to act immediately and take necessary remedial measures according to the nature of the intrusion.

Both flexibility and robustness can be achieved using an easy-to-understand and lightweight rule-description language that aids in writing simple Snort rules. Consider the following two primary principles while writing Snort rules:

- No written rule must extend beyond a single line; thus, rules should be short, precise, and easy to understand.
- Each rule should be divided into two logical sections:
  - The rule header
  - The rule options

The rule header contains the rule's action, the protocol, the source and destination IP addresses, the source and destination port information, and the **Classless Inter-Domain Routing (CIDR) block**. The rule option section includes alert messages in addition to information about the inspected part of the packet to determine whether to take any rule action.



```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg: "mountd access";)
```

Figure 12.20: Example of Snort rules

Snort Rules: Rule Actions and IP Protocols

The rule header stores a complete set of rules to identify a packet and determines the action to be performed or rule to be applied. It contains information that defines the who, where, and what of a packet, as well as what to do if a packet with all the attributes indicated in the rule should show up. The first item in a rule is the rule action, which tells Snort "what to do" when it finds a packet that matches the rule criteria. There are five available default actions in Snort: alert, log, pass, activate, and dynamic. Furthermore, if Snort is running in the inline mode, you have additional options, which include drop and reject.

The IP sends data from one system to another via the Internet. It supports unique addressing for every computer on a network. Organize data on the IP network into packets. Each packet contains message data, source, destination, and more.

Snort supports three available IP protocols to tackle suspicious behavior:

- **TCP**: The Transmission Control Protocol (TCP) is a part of the IP. It is used to connect two different hosts and exchange data between them.
- **UDP**: **The** User Datagram Protocol (UDP) is used for broadcasting messages over a network.
- **ICMP**: The Internet Control Message Protocol (ICMP) is a part of the IP. The OS uses ICMP in a network to send error messages, for example.

Snort Rules: Direction Operator and IP Addresses

- **Direction Operator**

  This operator indicates the direction of interest for the traffic; traffic can flow either in a single direction or bidirectionally.

  Example of a Snort rule using the Bidirectional Operator:

  ```
  log !192.168.1.0/24 any <> 192.168.1.0/24 23
  ```

- **IP Addresses**
  - Identify the IP address and port that the rule applies to
  - Use keyword "any" to define the IP address
  - Use numeric IP addresses qualified with a CIDR netmask
  - Example of IP Address Negation Rule:

    ```
    alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 111
    (content: "|00 01 86 a5|"; msg: "external mountd access";)
    ```

Snort Rules: Port Numbers

Port numbers can be listed in different ways, including the use of "any" ports, static port definitions, port ranges, and by negation. Port ranges are indicated by the range operator ":." The direction operator “-$>$” indicates the orientation or direction of the traffic to which the rule applies. Consider an IP address and port number on the left side of the direction operator as the traffic coming from the source host and the address and port information on the right side of the operator as the destination host. There is also a bidirectional operator, indicated by “$<>$”. This tells Snort to consider the address/port pairs in either the source or the destination orientation, and it is handy for recording/analyzing both sides of a conversation, such as telnet or POP3 sessions. Further, note that there is no “$<$-“ operator. In Snort versions before version 1.8.7, the direction operator did not provide proper error checking; hence, many people used invalid tokens. Note that “$<$-“ does not exist so that rules always read consistently.

The next fields in a Snort rule specify the source and destination IP addresses and ports of the packet, as well as the direction in which the packet is traveling. Snort can accept a single IP address or a list of addresses. When specifying a list of IP address, you should separate each one with a comma and then enclose the list within square brackets as follows:

[192.168.1.1,192.168.1.45,10.1.1.24]

When doing this, be careful not to use any whitespaces. You can also specify the ranges of IP addresses using CIDR notation or even include CIDR ranges within lists. Snort also allows you to apply the logical NOT operator (“!”) to an IP address or CIDR range to specify that the rule should match all but that address or range of addresses. For example, an easy modification to the initial example is to change it such that an alert is raised upon detecting any traffic that has originated outside the local net using the negation operator.

Example of a Port Negation:

```
log tcp any any -> 192.168.1.0/24 !6000:6010
```

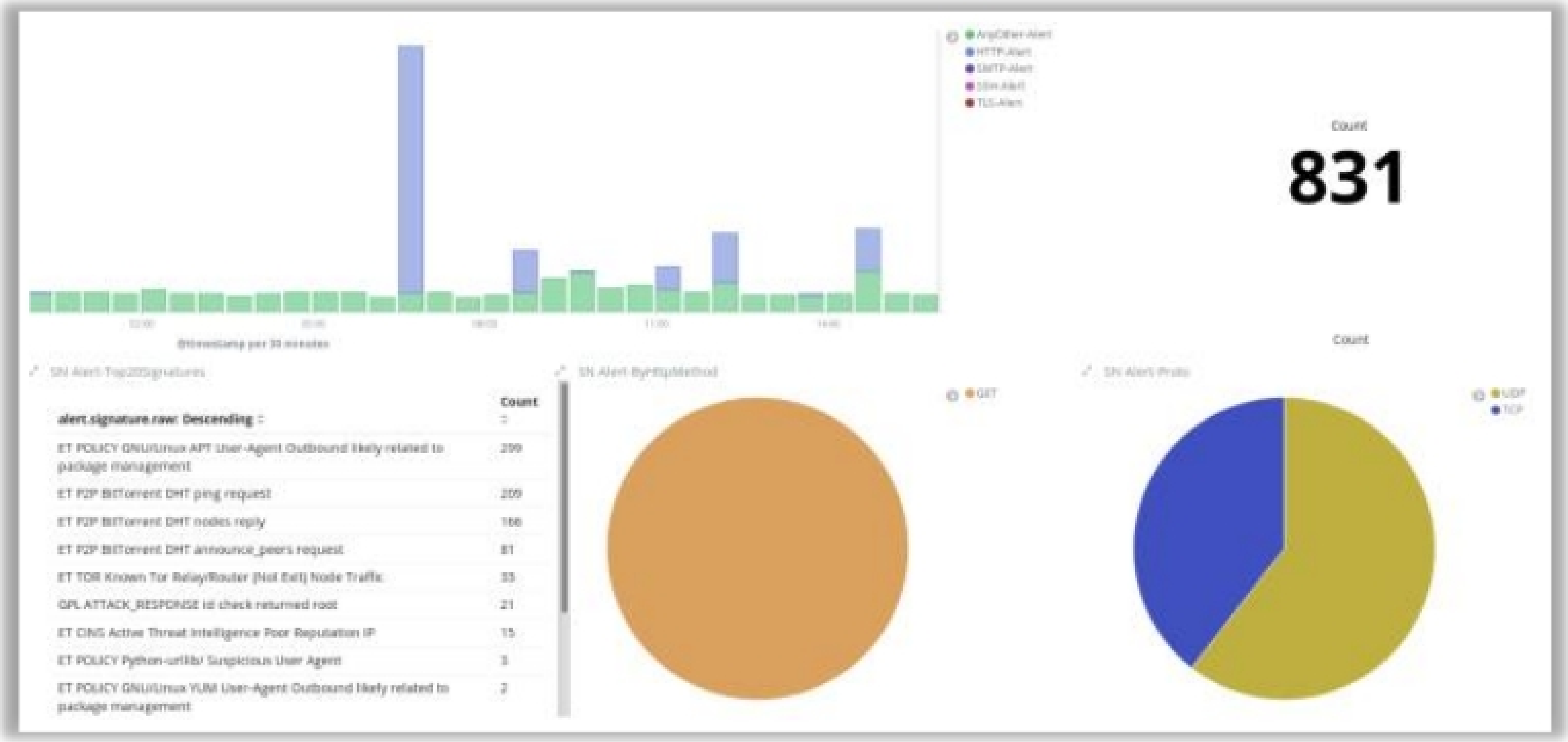
Protocols	IP Address	Action
Log UDP any any ->	192.168.1.0/24 1:1024	Log UDP traffic coming from any port and destination ports ranging from 1 to 1024
Log TCP any any ->	192.168.1.0/24 :5000	Log TCP traffic from any port going to ports less than or equal to 5000
Log TCP any :1024 ->	192.168.1.0/24 400:	Log TCP traffic from the well-known ports and going to ports greater than or equal to 400

Table 12.2: Examples of a Port Negation

- **Suricata**

  Source: *https://suricata-ids.org*

  Suricata is a robust network threat detection engine capable of real-time intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM), and offline pcap processing. It inspects the network traffic using powerful and extensive rules and a signature language, and it provides powerful Lua scripting support for the detection of complex threats. With standard input and output formats such as YAML and JSON, integrations with existing tools such as SIEMs, Splunk, Logstash/Elasticsearch, Kibana, and other databases become effortless.



alert.signature.raw: Descending	Count
ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management	299
ET P2P BitTorrent DHT ping request	209
ET P2P BitTorrent DHT nodes reply	166
ET P2P BitTorrent DHT announce_peers request	81
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic	33
GPL ATTACK_RESPONSE id check returned root	21
ET CINS Active Threat Intelligence Poor Reputation IP	15
ET POLICY Python-urllib/ Suspicious User Agent	3
ET POLICY GNU/Linux YUM User-Agent Outbound likely related to package management	2

Figure 12.21: Screenshot of TippingPoint

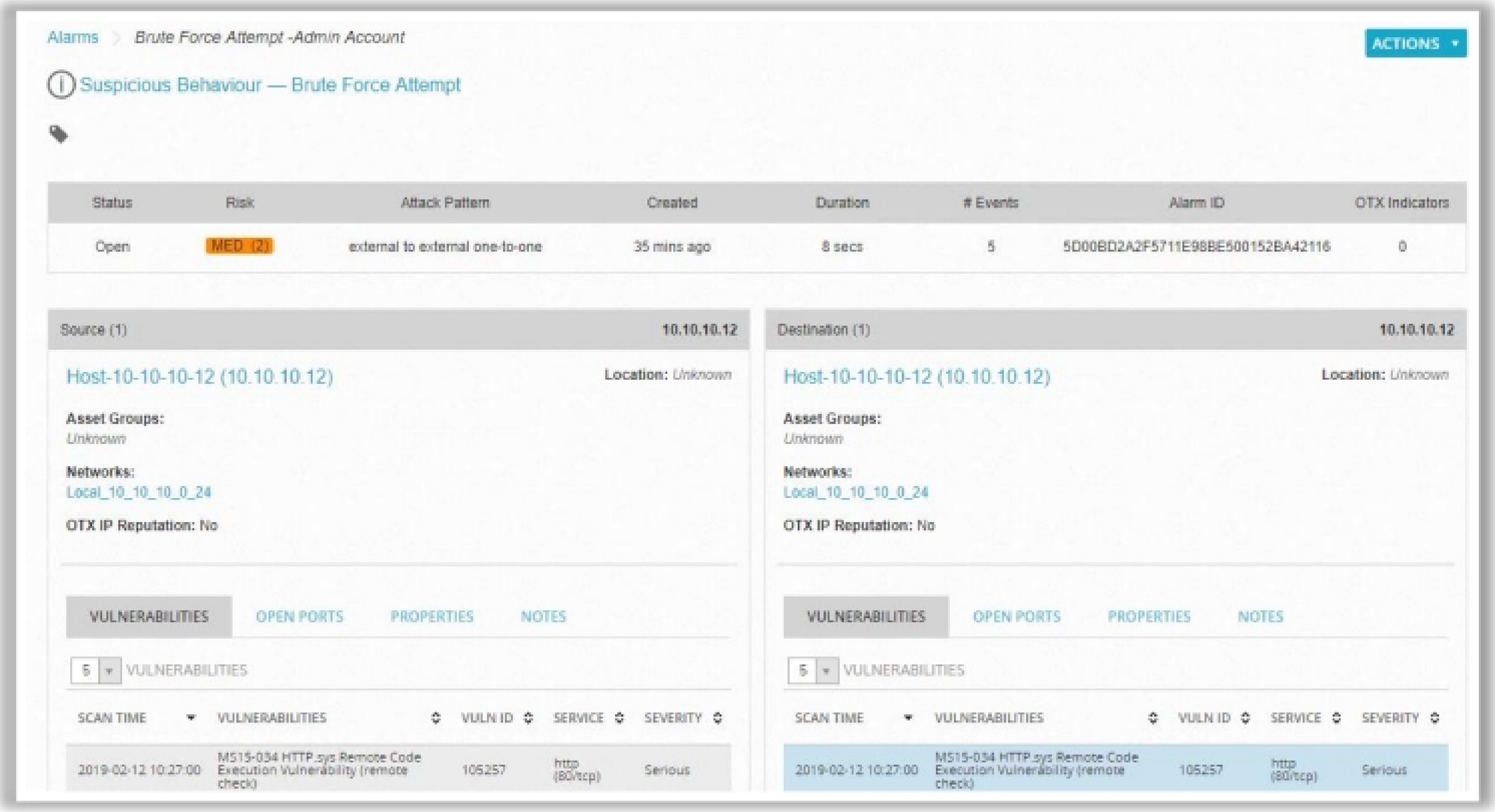
- **AlienVault® OSSIM™**

  Source: *https://www.alienvault.com*

  AlienVault® OSSIM™, Open Source Security Information and Event Management (SIEM), provides you with a feature-rich open-source SIEM complete with event collection, normalization, and correlation.

  OSSIM provides a unified platform with many essential security capabilities such as:

  - Asset discovery
  - Vulnerability assessment and intrusion detection
  - Behavioral monitoring
  - SIEM event correlation


Alarms
Brute Force Attempt -Admin Account
ACTIONS
Suspicious Behaviour — Brute Force Attempt

Status	Risk	Attack Pattern	Created	Duration	# Events	Alarm ID	OTX Indicators
Open	MED (2)	external to external one-to-one	35 mins ago	8 secs	5	5D00BD2A2F5711E98BE500152BA42116	0

Source (1) 10.10.10.12

Host-10-10-10-12 (10.10.10.12)
Location: Unknown
Asset Groups: Unknown
Networks: Local_10_10_10_0_24
OTX IP Reputation: No

Destination (1) 10.10.10.12

Host-10-10-10-12 (10.10.10.12)
Location: Unknown
Asset Groups: Unknown
Networks: Local_10_10_10_0_24
OTX IP Reputation: No

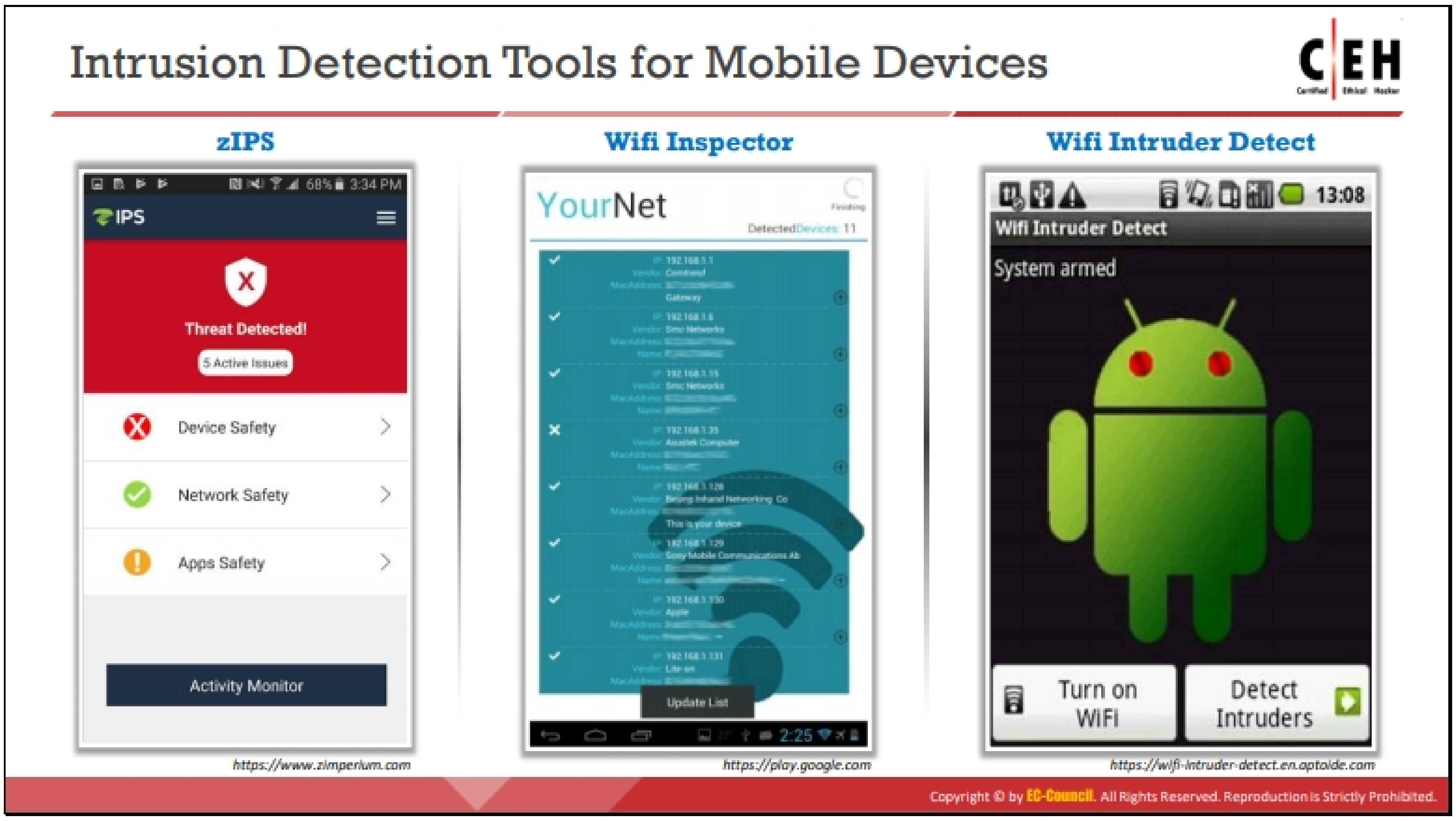
SCAN TIME	VULNERABILITIES	VULN ID	SERVICE	SEVERITY
2019-02-12 10:27:00	MS15-034 HTTP.sys Remote Code Execution Vulnerability (remote check)	105257	http (80/tcp)	Serious

SCAN TIME	VULNERABILITIES	VULN ID	SERVICE	SEVERITY
2019-02-12 10:27:00	MS15-034 HTTP.sys Remote Code Execution Vulnerability (remote check)	105257	http (80/tcp)	Serious

Figure 12.22: Screenshot of AlienVault® OSSIM™

Some additional intrusion detection tools are listed below:

- SolarWinds Security Event Manager (*https://www.solarwinds.com*)
- OSSEC (*https://www.ossec.net*)
- BroIDS/Zeek IDS (*https://www.zeek.org*)



Intrusion Detection Tools for Mobile Devices

zIPS

IPS
Threat Detected!
5 Active Issues
Device Safety
Network Safety
Apps Safety
Activity Monitor

https://www.zimperium.com

Wifi Inspector

YourNet
Update List

https://play.google.com

Wifi Intruder Detect

Wifi Intruder Detect
System armed
Turn on WiFi
Detect Intruders

https://wifi-intruder-detect.en.aptoide.com

Intrusion Detection Tools for Mobile Devices

Intrusion detection tools are also available for mobile devices to help you detect and prevent intrusion attempts.

- **zIPS**

  Source: *https://www.zimperium.com*

  Zimperium's zIPS™ is a mobile intrusion prevention system app that provides comprehensive protection for iOS and Android devices against mobile network, device, and application cyber-attacks. It can detect both known and unknown threats by analyzing the behavior of your mobile device. By examining slight deviations from the mobile device's OS statistics, memory, CPU, and other system parameters, the z9™ detection engine can accurately identify not only the specific type of malicious attack but also the forensics associated with the who, what, where, when, and how of an attack occurrence.

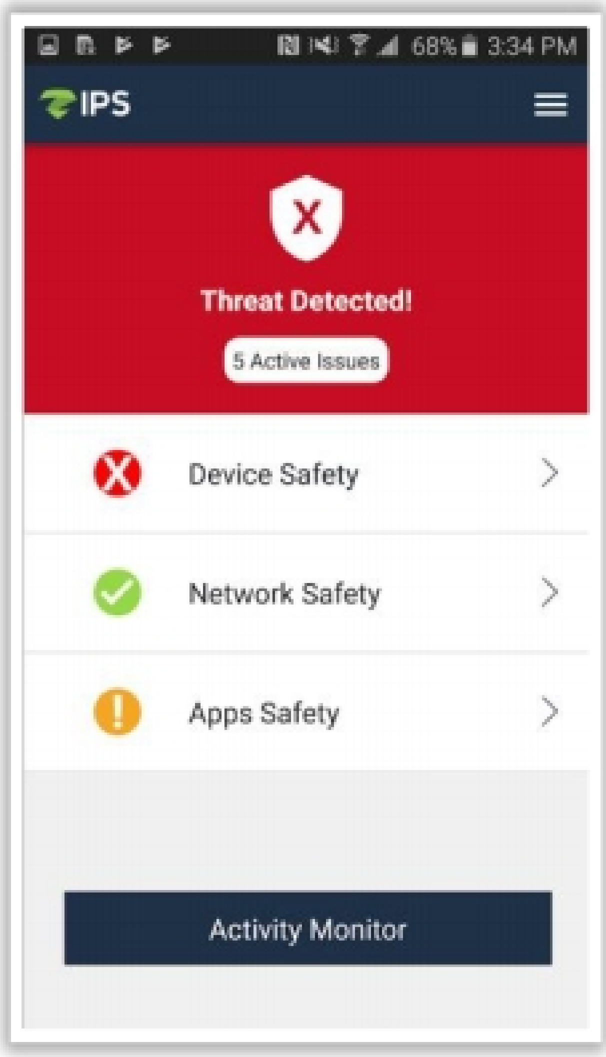
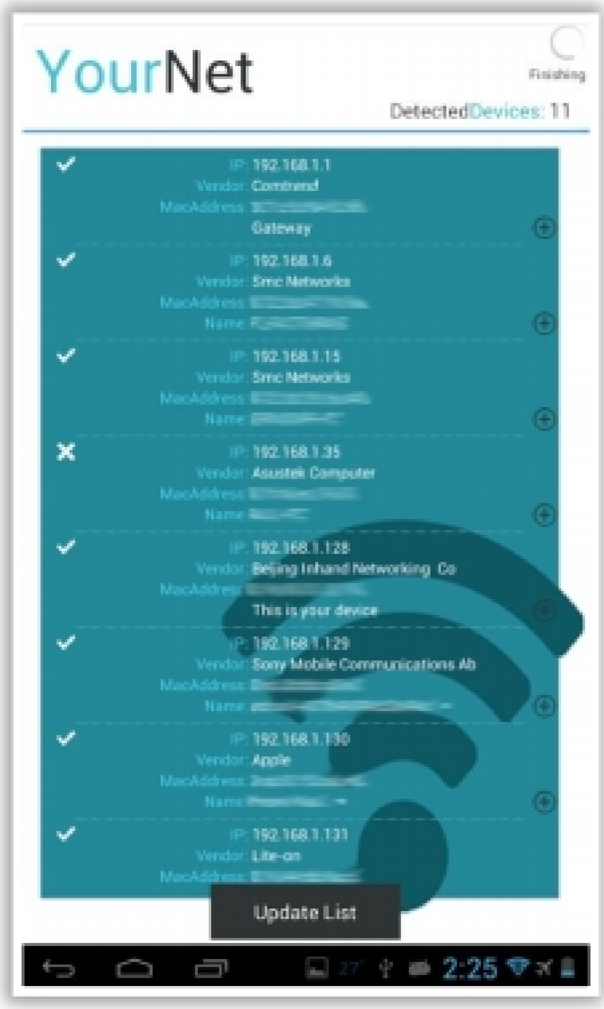

3:34 PM
68%
zIPS
Threat Detected!
5 Active Issues
Device Safety
Network Safety
Apps Safety
Activity Monitor

Figure 12.23: Screenshot of zIPS

- **Wifi Inspector**

  Source: *https://play.google.com*

  Wifi Inspector allows you to find all the devices connected to the network (via both wired and Wi-Fi connections, including consoles, TVs, PCs, tablets, and phones); it gives relevant data such as the IP addresses, manufacturer names, device names, and MAC addresses of connected devices. It also allows you to save a list of known devices with a custom name and finds intruders in a short period.


YourNet
Finishing
Update List
2:25

DetectedDevices: 11

IP: 192.168.1.1
Vendor: Comtrend
MacAddress: [redacted]
Gateway

IP: 192.168.1.6
Vendor: Smc Networks
MacAddress: [redacted]
Name: [redacted]

IP: 192.168.1.15
Vendor: Smc Networks
MacAddress: [redacted]
Name: [redacted]

IP: 192.168.1.35
Vendor: Asustek Computer
MacAddress: [redacted]
Name: [redacted]

IP: 192.168.1.128
Vendor: Beijing Inhand Networking Co
MacAddress: [redacted]
This is your device

IP: 192.168.1.129
Vendor: Sony Mobile Communications Ab
MacAddress: [redacted]
Name: [redacted]

IP: 192.168.1.130
Vendor: Apple
MacAddress: [redacted]
Name: [redacted]

IP: 192.168.1.131
Vendor: Lite-on
MacAddress: [redacted]

Figure 12.24: Screenshot of Wifi Inspector

- **Wifi Intruder Detect**

  Source: *https://wifi-intruder-detect.en.aptoide.com*

  Wifi Intruder Detect helps to find security leaks in the Wi-Fi network Internet connection. It allows you to detect an intruder who is accessing the network, Wi-Fi, or Internet connection without your consent.

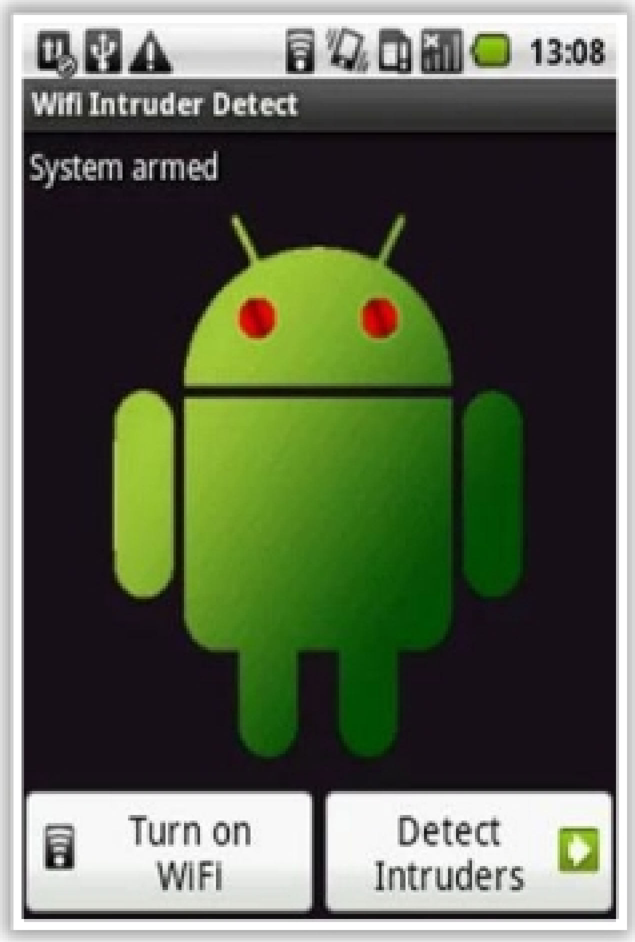
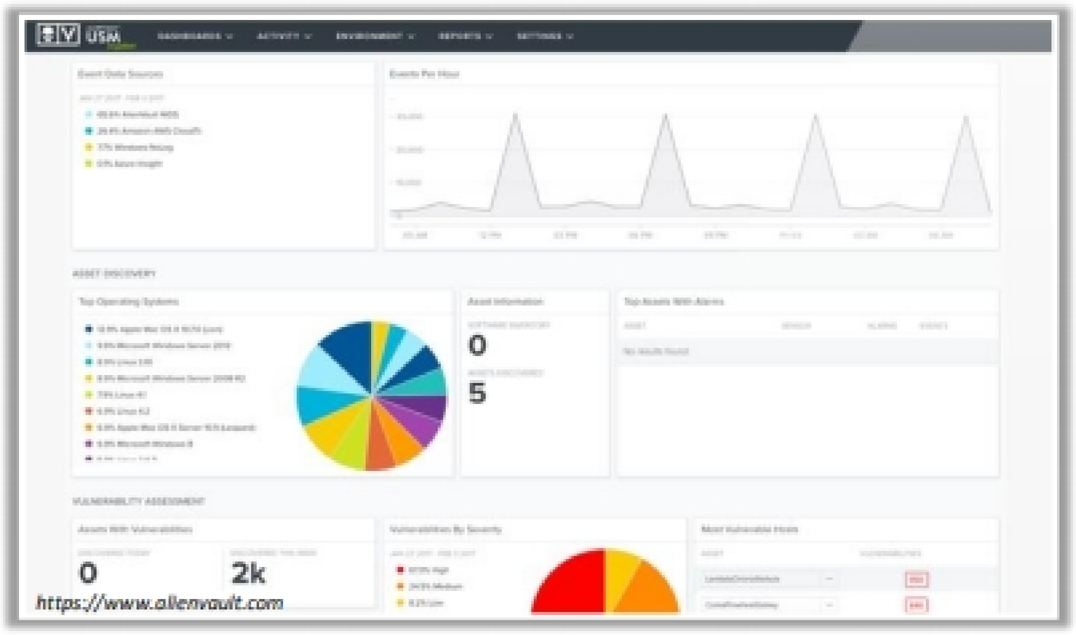

13:08
Wifi Intruder Detect
System armed
Turn on WiFi
Detect Intruders

Figure 12.25: Screenshot of Wifi Intruder Detector Pro

Intrusion Prevention Tools

AlienVault® Unified Security Management® (USM)

- AlienVault USM offers **threat detection**, **incident response**, and **compliance management** across the cloud, on-premises, and in hybrid environments
- It can be integrated with **AlienVault Open Threat Exchange (OTX)** to protect the network from intrusions



https://www.alienvault.com

IBM Security Network Intrusion Prevention System
https://www.ibm.com

Cyberoam Intrusion Prevention System
https://www.cyberoam.com

McAfee Host Intrusion Prevention for Desktops
https://www.mcafee.com

Cisco Intrusion Prevention Systems
https://www.cisco.com

Check Point IPS Software Blade
https://www.checkpoint.com

Intrusion Prevention Tools

- **AlienVault® Unified Security Management® (USM)**

  Source: *https://www.alienvault.com*

  AlienVault USM can perform threat detection, incident response, and compliance management across cloud, on-premises, and hybrid environments. It can be integrated with AlienVault Open Threat Exchange (OTX), which is an open threat intelligence community with more than 100,000 participants who contribute over 19 million threat indicators daily to protect the network from intrusions.

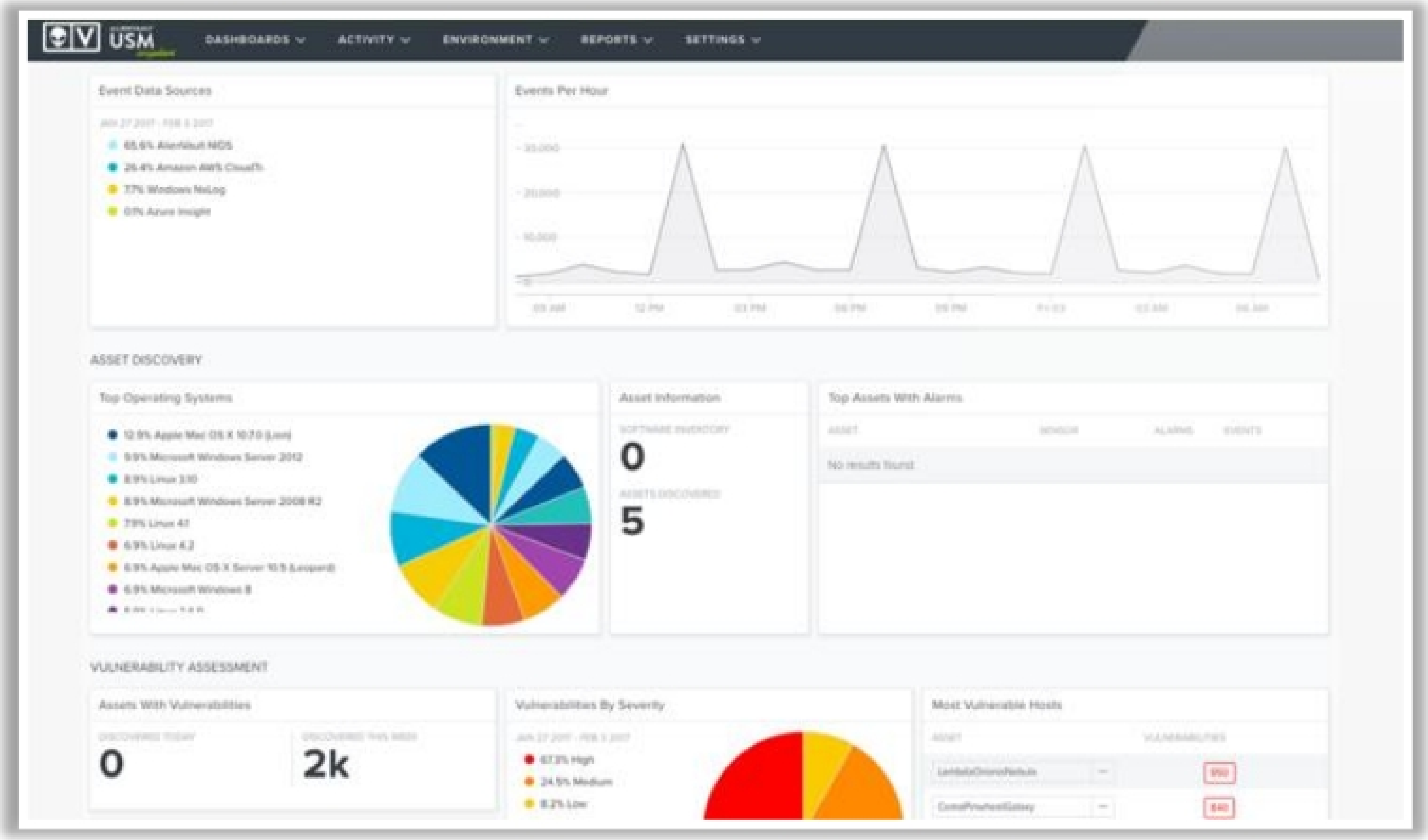

USM
DASHBOARDS
ACTIVITY
ENVIRONMENT
REPORTS
SETTINGS
Event Data Sources
Events Per Hour
ASSET DISCOVERY
Top Operating Systems
Asset Information
0
5
Top Assets With Alarms
No results found
VULNERABILITY ASSESSMENT
Assets With Vulnerabilities
0
2k
Vulnerabilities By Severity
Most Vulnerable Hosts

Figure 12.26: Screenshot of AlienVault USM

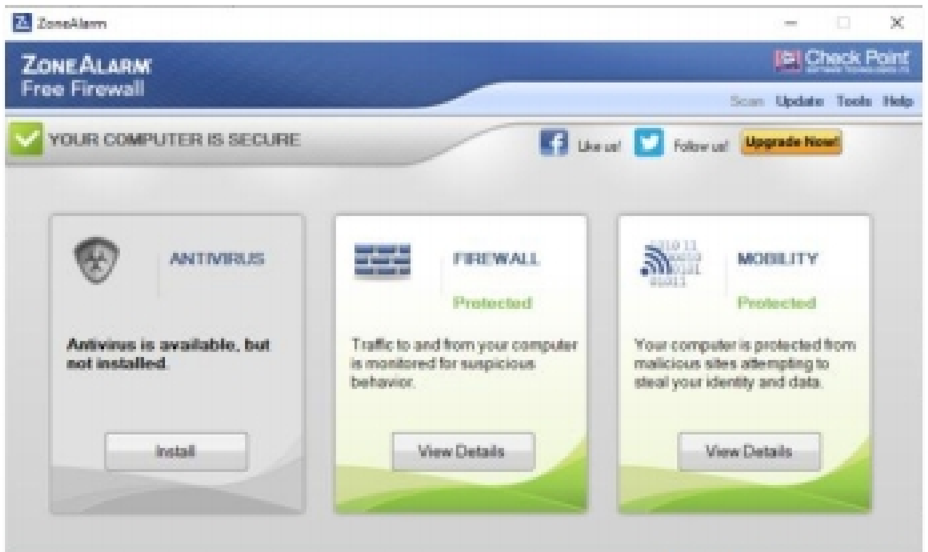
Some additional intrusion prevention tools are listed below:

- IBM Security Network Intrusion Prevention System (*https://www.ibm.com*)
- Cyberoam Intrusion Prevention System (*https://www.cyberoam.com*)
- McAfee Host Intrusion Prevention for Desktops (*https://www.mcafee.com*)
- Cisco Intrusion Prevention Systems (*https://www.cisco.com*)
- Check Point IPS Software Blade (*https://www.checkpoint.com*)

Firewalls: ZoneAlarm Free Firewall 2019 and ManageEngine Firewall Analyzer

ZoneAlarm Free Firewall 2019

ZoneAlarm Free Firewall 2019 **manages and monitors** all **incoming** and **outgoing** traffic and shields the network from hackers, malware, and other online threats that put **network privacy at risk**


ZoneAlarm
ZONEALARM
Free Firewall
Check Point
Scan
Update
Tools
Help
YOUR COMPUTER IS SECURE
Like us!
Follow us!
Upgrade Now!
ANTIVIRUS
FIREWALL
Protected
MOBILITY
Protected
Install
View Details
View Details

Antivirus is available, but not installed.

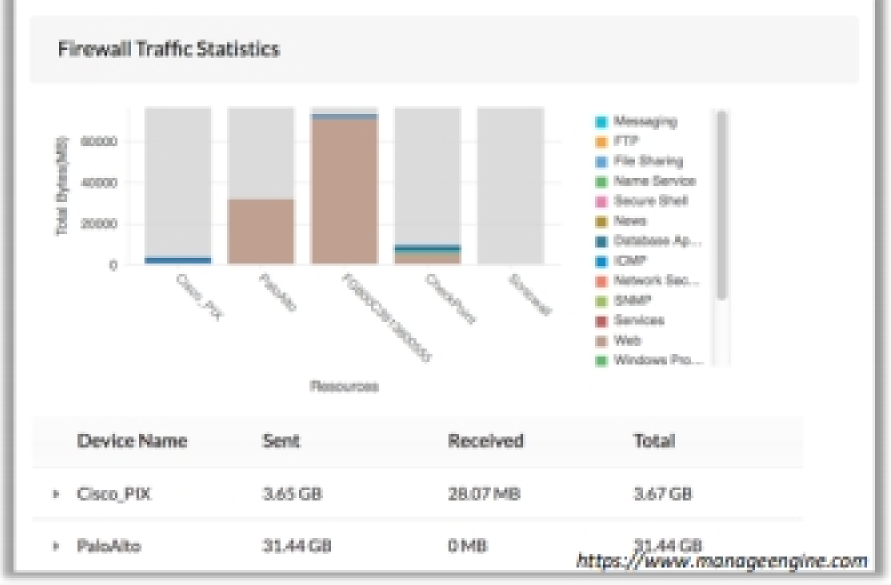
Traffic to and from your computer is monitored for suspicious behavior.

Your computer is protected from malicious sites attempting to steal your identity and data.

https://www.zonealarm.com

ManageEngine Firewall Analyzer

ManageEngine Firewall Analyzer offers a rich set of **pre-defined reports** that help in analyzing **bandwidth usage** and understanding **network security**


Firewall Traffic Statistics
Total Bytes(MB)
60000
40000
20000
0
Cisco_PIX
PaloAlto
FG800C3912800565
CheckPoint
Sonicwall
Resources
Messaging
FTP
File Sharing
Name Service
Secure Shell
News
Database Ap...
ICMP
Network Sec...
SNMP
Services
Web
Windows Pro...

Device Name	Sent	Received	Total
Cisco_PIX	3.65 GB	28.07 MB	3.67 GB
PaloAlto	31.44 GB	0 MB	31.44 GB

https://www.manageengine.com

Firewalls

- pfSense (*https://www.pfsense.org*)
- Sophos XG Firewall (*https://www.sophos.com*)
- Comodo Firewall (*https://personalfirewall.comodo.com*)
- Palo Alto Network Wildfire (*https://www.paloaltonetworks.com*)

Firewalls

Firewalls provide essential protection to computers against viruses, privacy threats, objectionable content, hackers, and malicious software when connected to the Internet. A firewall monitors running applications that access the network. It analyzes downloads, raises an alert when downloading a malicious file, and stops it from infecting a PC.

- **ZoneAlarm Free Firewall 2019**

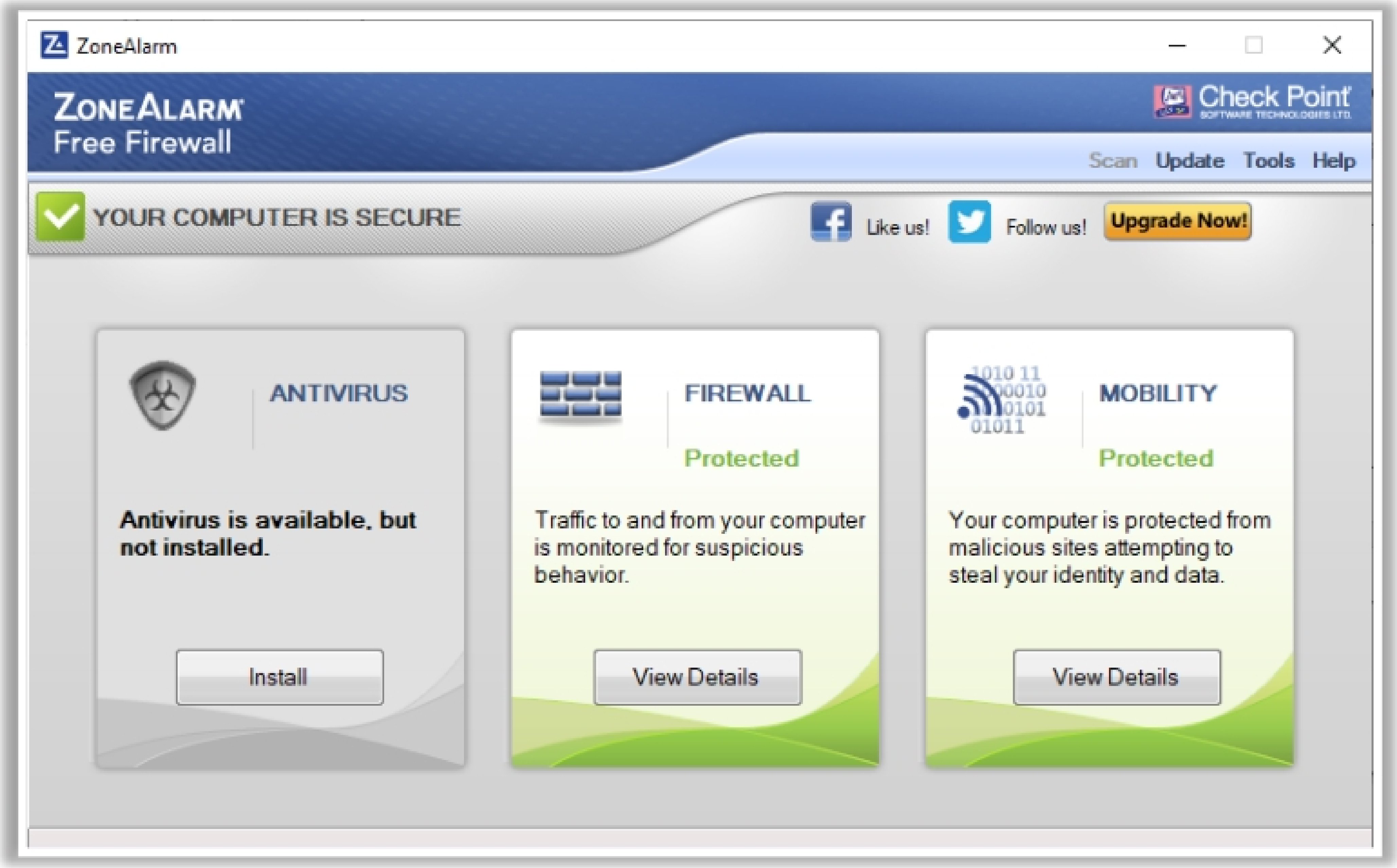
  Source: *https://www.zonealarm.com*

  ZoneAlarm Free Firewall 2019 prevents attackers and intruders from accessing your system. It manages and monitors all incoming and outgoing traffic and shields the network from hackers, malware, and other online threats that may compromise network privacy. It monitors programs for suspicious behavior, spotting and stopping new attacks that bypass traditional anti-virus protection. Moreover, it prevents identity theft by guarding your data. It also erases your tracks, allowing you to surf the web in complete privacy. Furthermore, it locks out attackers, blocks intrusions, and makes your PC invisible online. In addition, it filters out annoying and potentially dangerous emails.

  Features:

  - Two-way firewall that monitors and blocks inbound as well as outbound traffic
  - Allows users to browse the web privately using the Full Stealth Mode
  - Identity protection services help to prevent identify theft by guarding crucial data of the users. It also offers PC protection and data encryption
  - Public network protection and wireless network protection are other key features of this firewall

- Provides quick real-time security updates


ZoneAlarm
ZONEALARM
Free Firewall
Check Point
SOFTWARE TECHNOLOGIES LTD.
Scan
Update
Tools
Help
YOUR COMPUTER IS SECURE
Like us!
Follow us!
Upgrade Now!
ANTIVIRUS
Install
FIREWALL
Protected
View Details
MOBILITY
Protected
View Details

Antivirus is available, but not installed.

Traffic to and from your computer is monitored for suspicious behavior.

Your computer is protected from malicious sites attempting to steal your identity and data.

Figure 12.27: Screenshot of ZoneAlarm PRO FIREWALL 2017

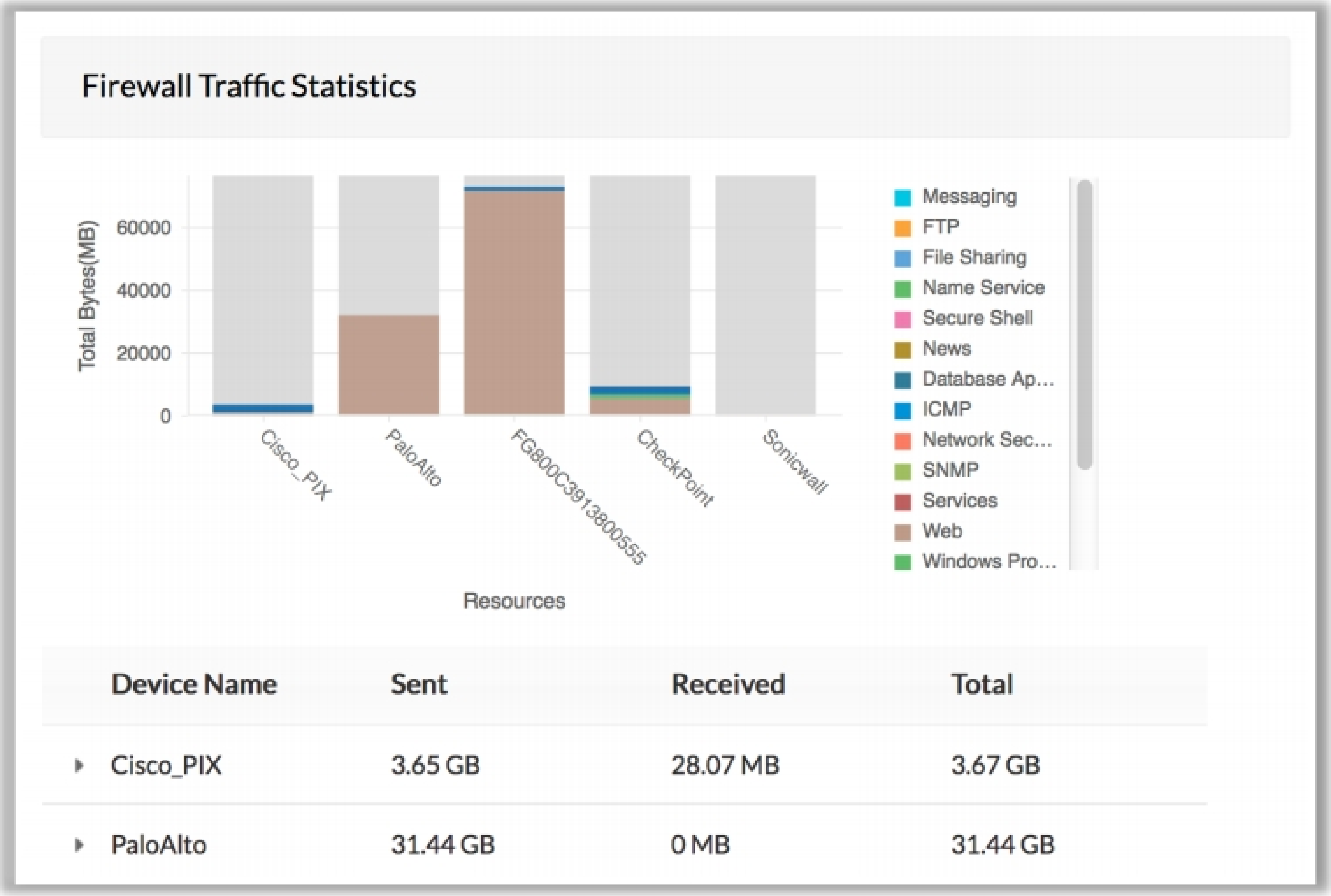
- **ManageEngine Firewall Analyzer**

  Source: *https://www.manageengine.com*

  ManageEngine Firewall Analyzer is an agent-less log analytics and configuration management software that helps network administrators to understand how the bandwidth is being used in their network. ManageEngine Firewall Analyzer is vendor-agnostic and supports nearly all open-source and commercial network firewalls such as Check Point, Cisco, Juniper, Fortinet, and Palo Alto.

  Features:

  - Compliance and Change Management
  - User Internet Activity Monitoring
  - Network Traffic and Bandwidth Monitoring
  - Firewall Policy Management
  - Real-time VPN and Proxy Server Monitoring
  - Network Security Management
  - Network Forensic Audits
  - Log Analysis

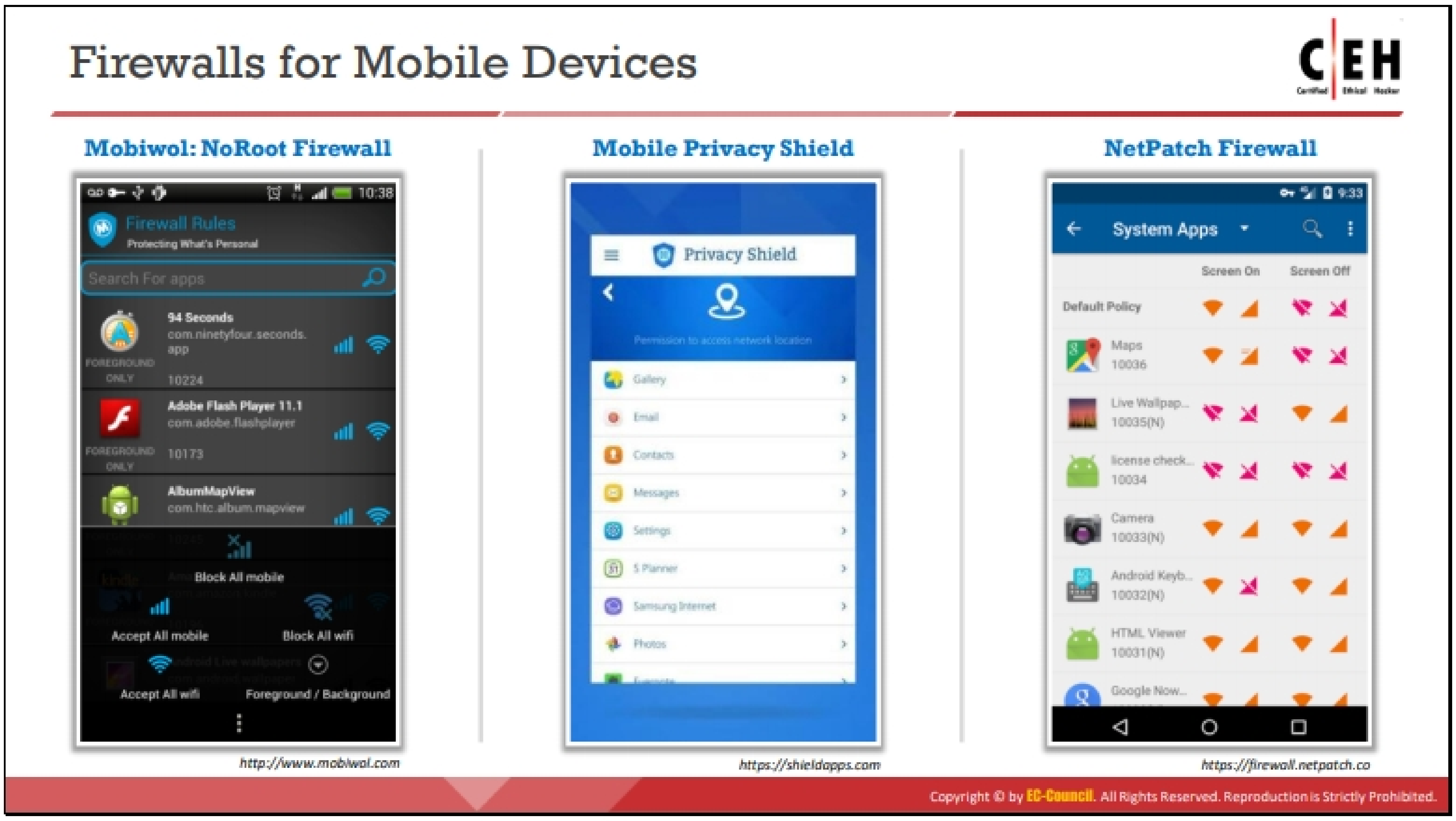

Firewall Traffic Statistics
Total Bytes(MB)
60000
40000
20000
0
Cisco_PIX
PaloAlto
FG800C3913800555
CheckPoint
Sonicwall
Resources
Messaging
FTP
File Sharing
Name Service
Secure Shell
News
Database Ap...
ICMP
Network Sec...
SNMP
Services
Web
Windows Pro...

Device Name	Sent	Received	Total
▸ Cisco_PIX	3.65 GB	28.07 MB	3.67 GB
▸ PaloAlto	31.44 GB	0 MB	31.44 GB

Figure 12.28: Screenshot of ManageEngine Firewall Analyzer

Some additional firewall solutions are listed below:

- pfSense (*https://www.pfsense.org*)
- Sophos XG Firewall (*https://www.sophos.com*)
- Comodo Firewall (*https://personalfirewall.comodo.com*)
- Palo Alto Network Wildfire (*https://www.paloaltonetworks.com*)



Firewalls for Mobile Devices

Mobiwol: NoRoot Firewall

Mobile Privacy Shield

NetPatch Firewall

Firewall Rules
Protecting What's Personal
Search For apps
94 Seconds
Adobe Flash Player 11.1
AlbumMapView
Block All mobile
Accept All mobile
Block All wifi
Accept All wifi
Foreground / Background
Privacy Shield
Permission to access network location
Gallery
Email
Contacts
Messages
Settings
S Planner
Samsung Internet
Photos
System Apps
Screen On
Screen Off
Default Policy
Maps
Live Wallpap...
license check...
Camera
Android Keyb...
HTML Viewer
Google Now...

http://www.mobiwol.com

https://shieldapps.com

https://firewall.netpatch.co

Firewalls for Mobile Devices

The firewalls discussed previously are used for securing personal computers and networks. Similarly, some firewalls can secure mobile devices.

- **Mobiwol: NoRoot Firewall**

  Source: *http://www.mobiwol.com*

  Mobiwol NoRoot Firewall helps to take control of mobile apps, easily allow/block app connectivity, and block background app activity. It generates alerts when new apps access the Internet.

  Features:

  - Automatic launches on device startup
  - Automatically identifies applications currently installed on your mobile device
  - Identifies and notifies when newly installed apps access the web
  - Sets allow/block on a per-application basis
  - Disables background activity for selected apps

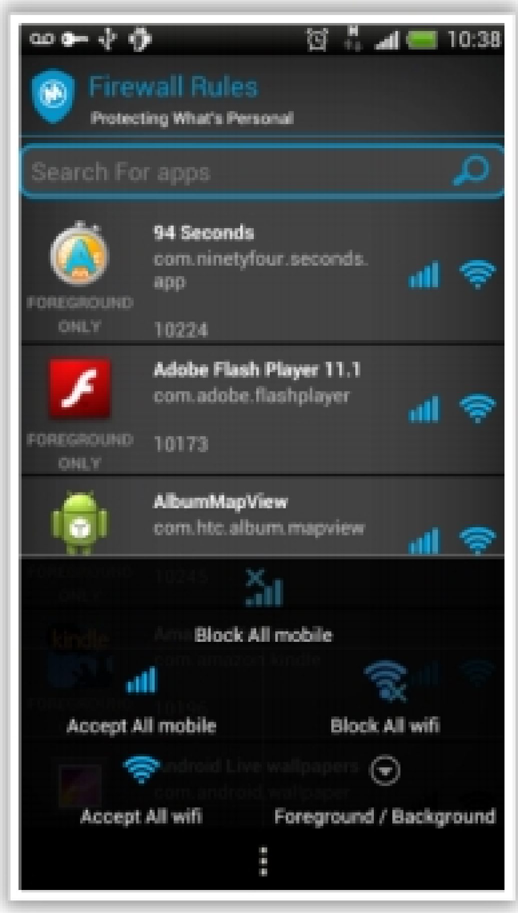

10:38
Firewall Rules
Protecting What's Personal
Search For apps
94 Seconds
com.ninetyfour.seconds.app
FOREGROUND ONLY
10224
Adobe Flash Player 11.1
com.adobe.flashplayer
FOREGROUND ONLY
10173
AlbumMapView
com.htc.album.mapview
Block All mobile
Accept All mobile
Block All wifi
Accept All wifi
Foreground / Background

Figure 12.29: Screenshot of Mobiwol: NoRoot Firewall

- **Mobile Privacy Shield**

  Source: *https://shieldapps.com*

  Mobile Privacy Shield is an application for people on the move, i.e., people who store necessary information on their smartphones and use their devices for banking, shopping, business, and more. Mobile Privacy Shield's Privacy Advisor monitors application permissions, sorting them into three categories by the privacy-risk level. Each report is packed with detailed information and a response is suggested per case. Mobile Privacy Shield centralizes all permissions, allowing you to review and assess their validity and need conveniently. It also allows you to remove each threat from within the interface.

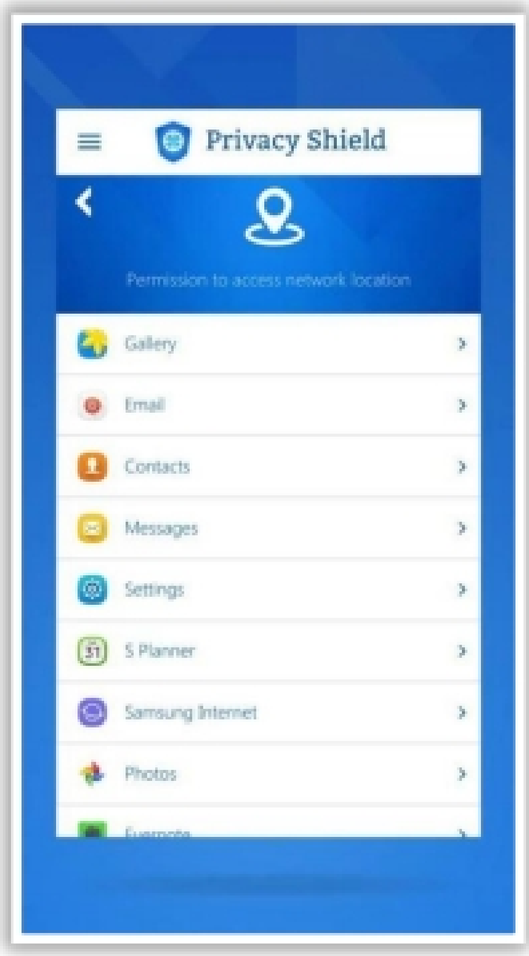

Privacy Shield
Permission to access network location
Gallery
Email
Contacts
Messages
Settings
S Planner
Samsung Internet
Photos
Evernote

Figure 12.30: Screenshot of Mobile Privacy Shield

- **NetPatch Firewall**

  Source: *https://firewall.netpatch.co*

  NetPatch Firewall is a full-featured advanced Android no-root firewall. It can be used to fully control a mobile device network. Using NetPatch Firewall, you can create network rules based on apps, IP addresses, domain names, etc. This firewall is designed to reduce a mobile device's network traffic and battery consumption, improve network security, and ensure privacy.

  Features:

  - Block network access per app, screen on/off, Wi-Fi/mobile (3G & 4G), block roaming
  - Shadowsocks secure proxy, support TCP and UDP (a better VPN proxy)
  - Custom DNS, change your DNS server, support DNS query through Shadowsocks proxy, and set the DNS cache time
  - Notify when new apps installed
  - Export/import configure

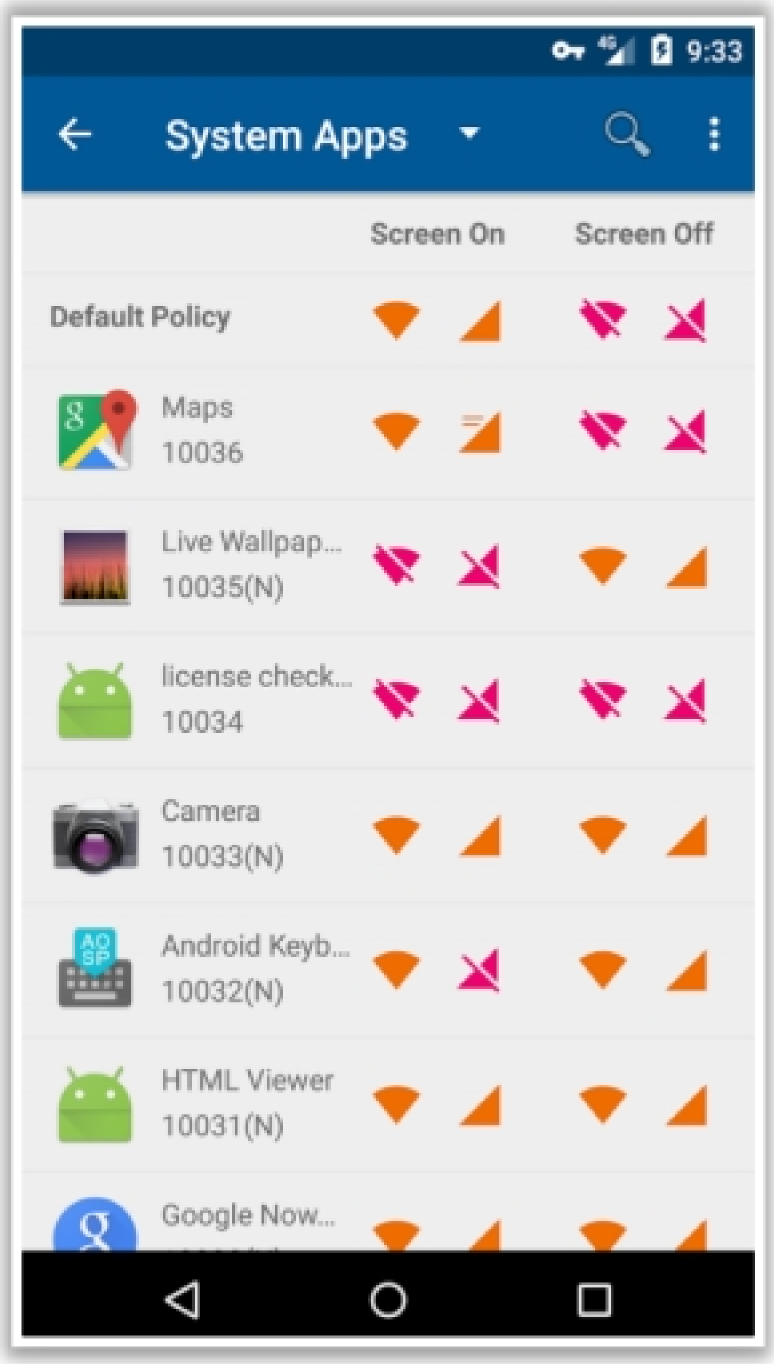
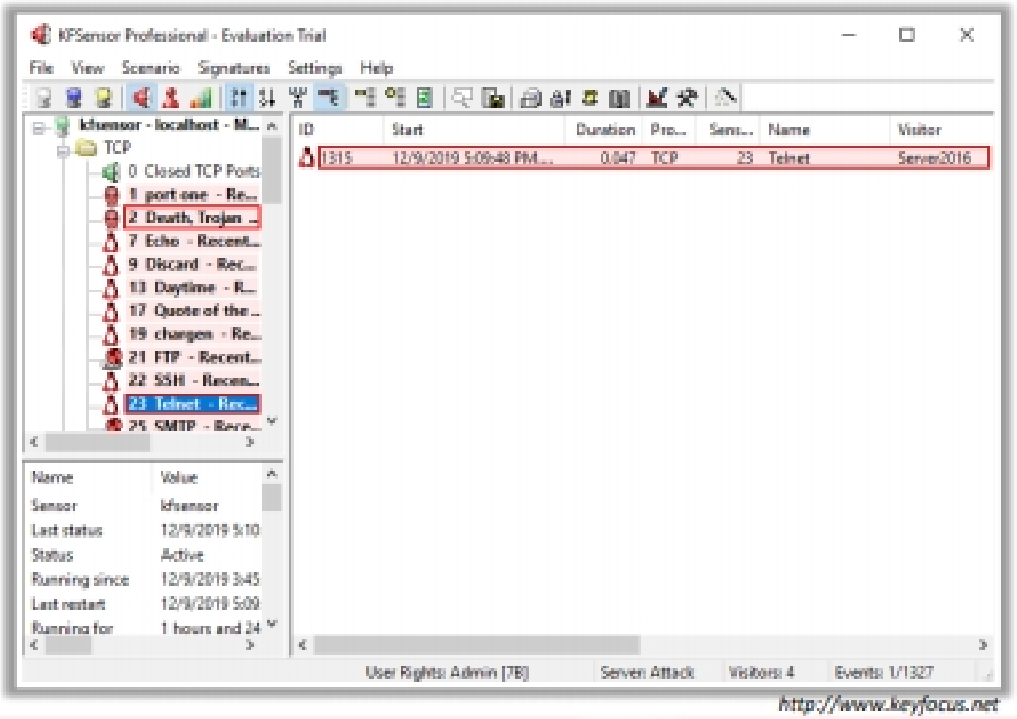

9:33
System Apps
Screen On
Screen Off
Default Policy
Maps
10036
Live Wallpap...
10035(N)
license check...
10034
Camera
10033(N)
Android Keyb...
10032(N)
HTML Viewer
10031(N)
Google Now...

Figure 12.31: Screenshot of NetPatch Firewall

Honeypot Tools: KFSensor and SPECTER

KFSensor

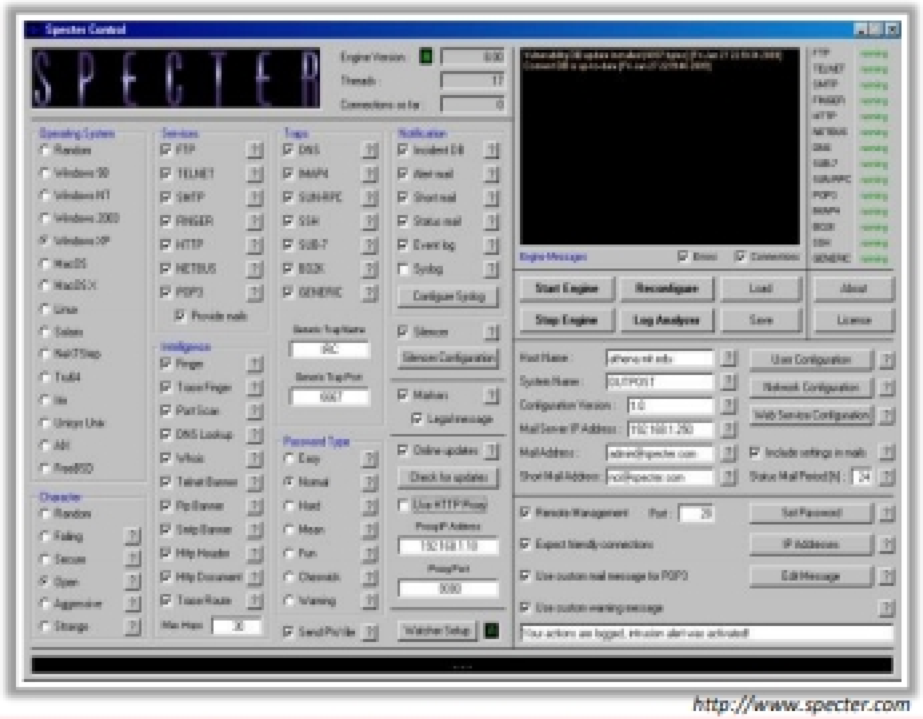
KFSensor is a **host-based** Intrusion Detection System (IDS) that acts as a honeypot to attract and detect hackers and worms by **simulating vulnerable system services** and **Trojans**


KFSensor Professional - Evaluation Trial
File View Scenario Signatures Settings Help
kfsensor - localhost
TCP
0 Closed TCP Ports
1 port one
2 Death, Trojan
7 Echo - Recent
9 Discard - Rec
13 Daytime - R
17 Quote of the
19 chargen - Re
21 FTP - Recent
22 SSH - Recen
23 Telnet - Rec
25 SMTP - Rece
ID Start Duration Pro... Sens... Name Visitor
1315 12/9/2019 5:09:48 PM... 0.047 TCP 23 Telnet Server2016
Name Value
Sensor kfsensor
Last status 12/9/2019 5:10
Status Active
Running since 12/9/2019 3:45
Last restart 12/9/2019 5:09
Running for 1 hours and 24
User Rights: Admin [78] Server: Attack Visitors: 4 Events: 1/1327

http://www.keyfocus.net

SPECTER

SPECTER is a **honeypot-based** intrusion detection system that offers common **Internet services** such as SMTP, FTP, POP3, HTTP, and TELNET which appear perfectly normal to the attackers but in fact are traps


Specter Control
SPECTER
Start Engine
Stop Engine
Reconfigure
Log Analyzer
Load
Save
About
License

http://www.specter.com

Honeypot Tools

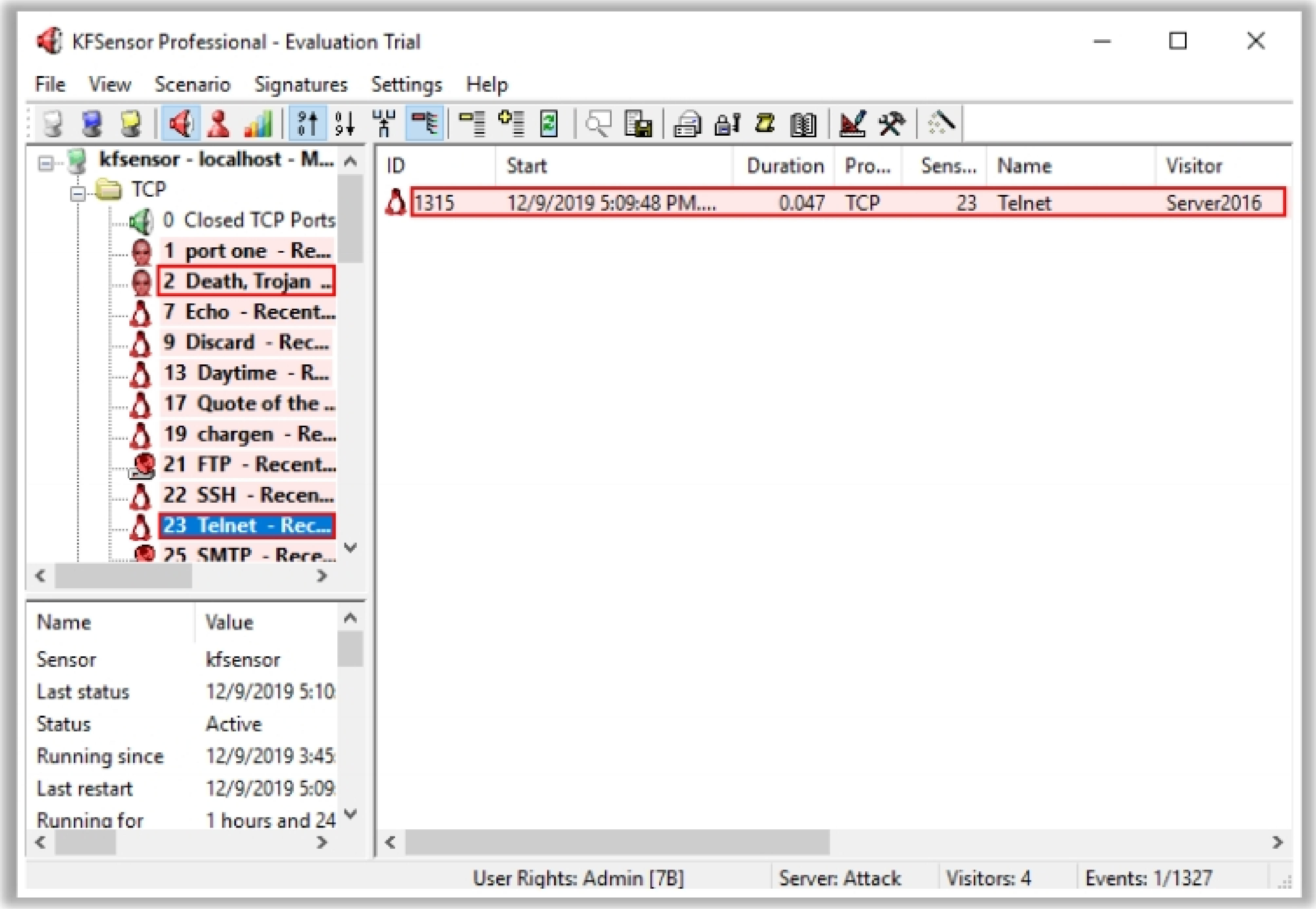
Honeypots are security tools that allow the security community to monitor attackers' tricks and exploits by logging all their activity so that it can respond to such exploits quickly before the attacker can misuse or compromise the system.

- **KFSensor**

  Source: *http://www.keyfocus.net*

  KFSensor is a host-based IDS that acts as a honeypot to attract and detect hackers and worms by simulating vulnerable system services and Trojans. By acting as a decoy server, it can divert attacks from critical systems and provide a higher level of information than that achieved using firewalls and NIDS alone.

  You can use KFSensor in a Windows-based corporate environment. It includes many innovative and unique features such as remote management, a Snort-compatible signature engine, and emulations of Windows networking protocols.


KFSensor Professional - Evaluation Trial
File View Scenario Signatures Settings Help
kfsensor - localhost - M...
TCP
0 Closed TCP Ports
1 port one - Re...
2 Death, Trojan ...
7 Echo - Recent...
9 Discard - Rec...
13 Daytime - R...
17 Quote of the ...
19 chargen - Re...
21 FTP - Recent...
22 SSH - Recen...
23 Telnet - Rec...
25 SMTP - Rece...
User Rights: Admin [7B]
Server: Attack
Visitors: 4
Events: 1/1327

ID	Start	Duration	Pro...	Sens...	Name	Visitor
1315	12/9/2019 5:09:48 PM....	0.047	TCP	23	Telnet	Server2016

Name	Value
Sensor	kfsensor
Last status	12/9/2019 5:10
Status	Active
Running since	12/9/2019 3:45
Last restart	12/9/2019 5:09
Running for	1 hours and 24

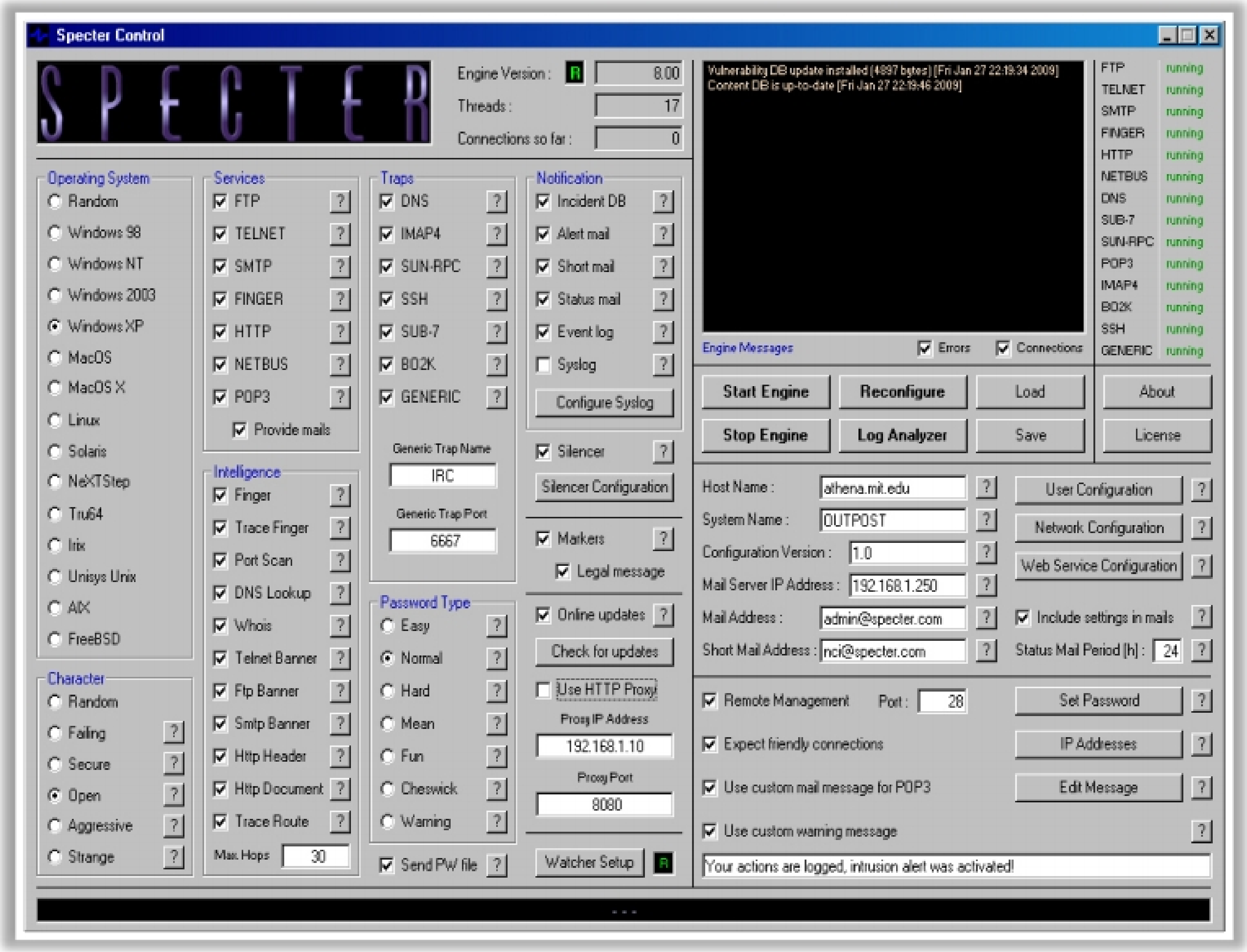
Figure 12.32: Screenshot of KFSensor

- **SPECTER**

  Source: *http://www.specter.com*

  SPECTER is a honeypot or deception system. It simulates a complete system and provides an appealing target to lure hackers away from production systems. It offers typical Internet services such as SMTP, FTP, POP3, HTTP, and TELNET, which appear perfectly normal to attackers. However, it traps attackers by tricking them into leaving some traces that show that they had connected to a decoy system that does none of the things it appears to but instead logs everything and notifies the appropriate people.

  Furthermore, SPECTER automatically investigates attackers while they are still trying to break in. It provides massive amounts of decoy content and generates decoy programs that do not leave traces on the attacker's computer. Automated weekly online updates of the honeypot's content and vulnerability databases allow the honeypot to change regularly without user interaction.


Specter Control
SPECTER
Start Engine
Reconfigure
Load
About
Stop Engine
Log Analyzer
Save
License
Configure Syslog
Silencer Configuration
Check for updates
Watcher Setup
User Configuration
Network Configuration
Web Service Configuration
Set Password
IP Addresses
Edit Message
Engine Messages
?

Engine Version : 8.00
Threads : 17
Connections so far : 0

Vulnerability DB update installed (4897 bytes) [Fri Jan 27 22:19:34 2009]
Content DB is up-to-date [Fri Jan 27 22:19:46 2009]

☑ Errors ☑ Connections

FTP running
TELNET running
SMTP running
FINGER running
HTTP running
NETBUS running
DNS running
SUB-7 running
SUN-RPC running
POP3 running
IMAP4 running
BO2K running
SSH running
GENERIC running

Operating System
○ Random
○ Windows 98
○ Windows NT
○ Windows 2003
● Windows XP
○ MacOS
○ MacOS X
○ Linux
○ Solaris
○ NeXTStep
○ Tru64
○ Irix
○ Unisys Unix
○ AIX
○ FreeBSD

Character
○ Random
○ Failing
○ Secure
● Open
○ Aggressive
○ Strange

Services
☑ FTP
☑ TELNET
☑ SMTP
☑ FINGER
☑ HTTP
☑ NETBUS
☑ POP3
☑ Provide mails

Intelligence
☑ Finger
☑ Trace Finger
☑ Port Scan
☑ DNS Lookup
☑ Whois
☑ Telnet Banner
☑ Ftp Banner
☑ Smtp Banner
☑ Http Header
☑ Http Document
☑ Trace Route
Max Hops 30

Traps
☑ DNS
☑ IMAP4
☑ SUN-RPC
☑ SSH
☑ SUB-7
☑ BO2K
☑ GENERIC
Generic Trap Name IRC
Generic Trap Port 6667

Password Type
○ Easy
● Normal
○ Hard
○ Mean
○ Fun
○ Cheswick
○ Warning
☑ Send PW file

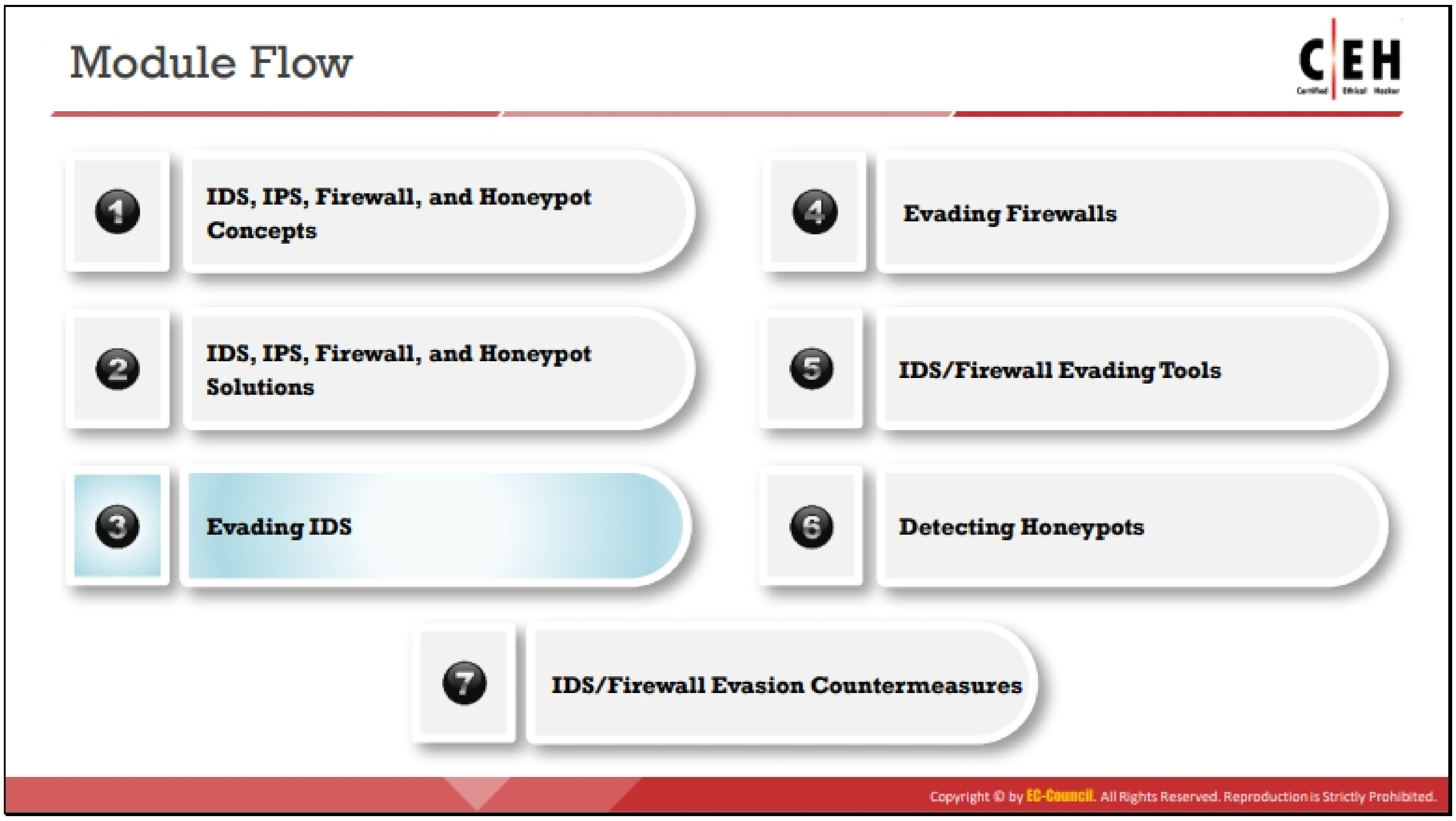
Notification
☑ Incident DB
☑ Alert mail
☑ Short mail
☑ Status mail
☑ Event log
☐ Syslog
☑ Silencer
☑ Markers
☑ Legal message
☑ Online updates
☐ Use HTTP Proxy
Proxy IP Address 192.168.1.10
Proxy Port 8080

Host Name : athena.mit.edu
System Name : OUTPOST
Configuration Version : 1.0
Mail Server IP Address : 192.168.1.250
Mail Address : admin@specter.com
Short Mail Address : nci@specter.com
☑ Include settings in mails
Status Mail Period [h] : 24
☑ Remote Management Port : 28
☑ Expect friendly connections
☑ Use custom mail message for POP3
☑ Use custom warning message
Your actions are logged, intrusion alert was activated!

Figure 12.33: Screenshot of SPECTER

Some additional honeypot tools are listed below:

- HoneyBOT (*https://www.atomicsoftwaresolutions.com*)
- MongoDB-HoneyProxy (*https://github.com*)
- Modern Honey Network (*https://github.com*)
- Honeyd (*http://www.honeyd.org*)

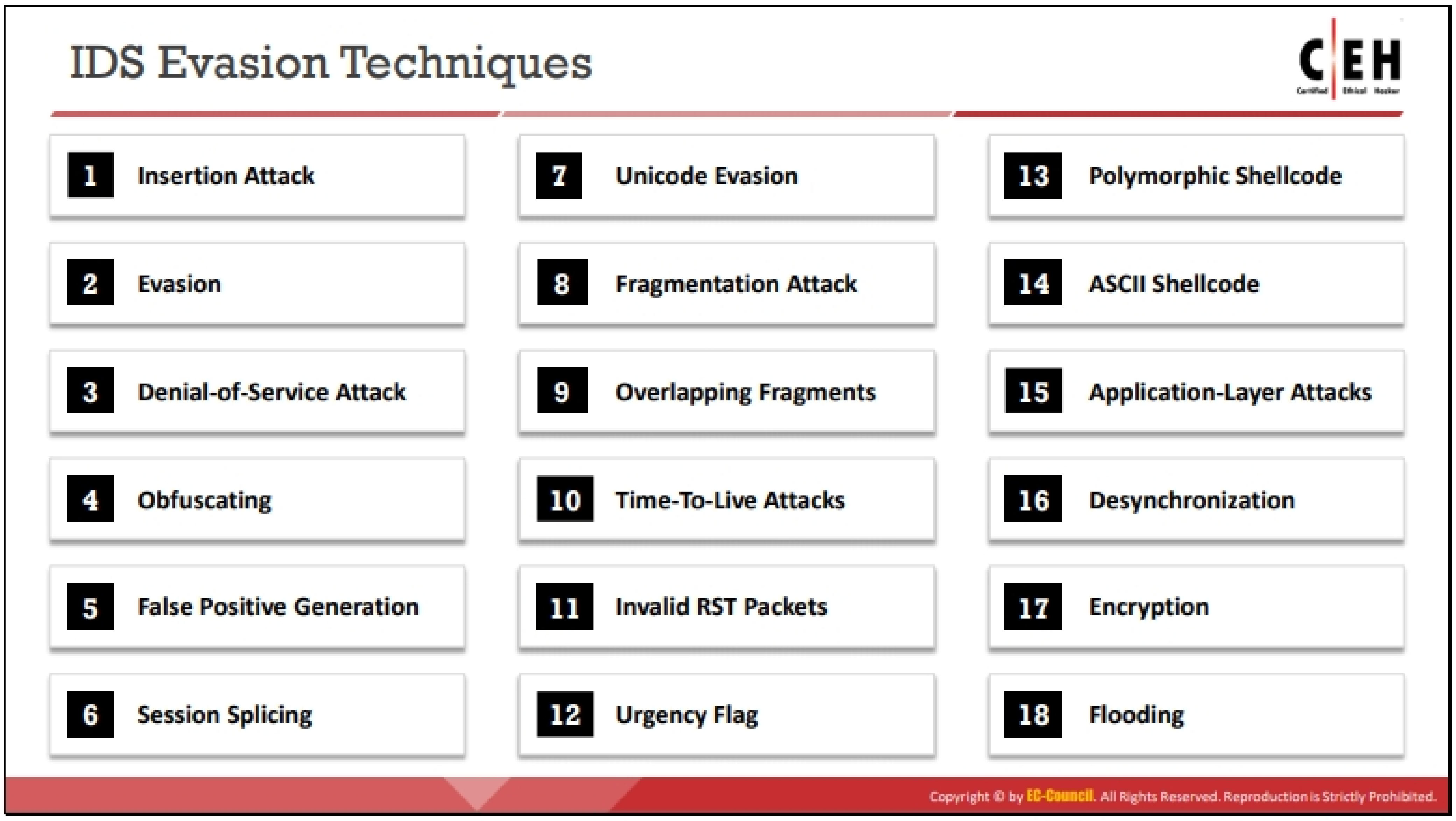


Module Flow

1. IDS, IPS, Firewall, and Honeypot Concepts
2. IDS, IPS, Firewall, and Honeypot Solutions
3. Evading IDS
4. Evading Firewalls
5. IDS/Firewall Evading Tools
6. Detecting Honeypots
7. IDS/Firewall Evasion Countermeasures

Evading IDS

The previous sections helped us to understand IDS, IPS, their roles and functions, how they protect your network from intruders, and the various IDS solutions available. Even though IDS thwart attempts to breach the network security, attackers can still evade IDS. This section explains various ways in which attackers evade IDS.



IDS Evasion Techniques

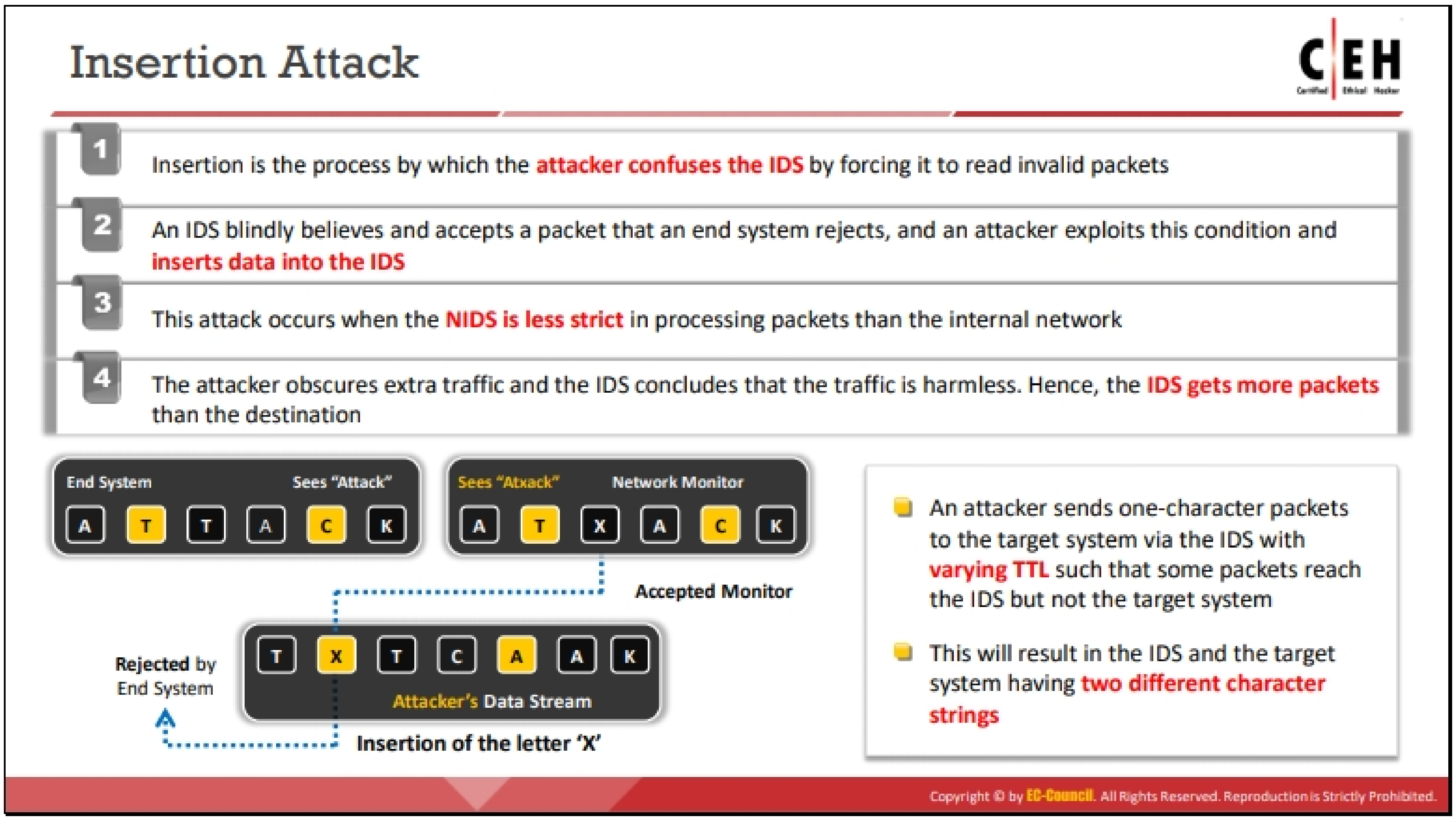
1. Insertion Attack
2. Evasion
3. Denial-of-Service Attack
4. Obfuscating
5. False Positive Generation
6. Session Splicing
7. Unicode Evasion
8. Fragmentation Attack
9. Overlapping Fragments
10. Time-To-Live Attacks
11. Invalid RST Packets
12. Urgency Flag
13. Polymorphic Shellcode
14. ASCII Shellcode
15. Application-Layer Attacks
16. Desynchronization
17. Encryption
18. Flooding

IDS Evasion Techniques

IDS that provide an extra layer of security to the organization's infrastructure are interesting targets for attackers. Attackers implement various IDS evasion techniques to bypass such security mechanisms and compromise the infrastructure. IDS evasion is the process of modifying attacks to fool the IDS/IPS into interpreting that the traffic is legitimate and thus prevent the IDS from triggering an alert. Many IDS evasion techniques can perform IDS evasion in different and effective ways.

Some IDS evasion techniques are as follows

- Insertion Attack
- Evasion
- DoS Attack
- Obfuscating
- False Positive Generation
- Session Splicing
- Unicode Evasion
- Fragmentation Attack
- Overlapping Fragments
- Time-To-Live Attacks
- Invalid RST Packets
- Urgency Flag
- Polymorphic Shellcode
- ASCII Shellcode
- Application-Layer Attacks
- Desynchronization
- Encryption
- Flooding



Insertion Attack

1. Insertion is the process by which the **attacker confuses the IDS** by forcing it to read invalid packets
2. An IDS blindly believes and accepts a packet that an end system rejects, and an attacker exploits this condition and **inserts data into the IDS**
3. This attack occurs when the **NIDS is less strict** in processing packets than the internal network
4. The attacker obscures extra traffic and the IDS concludes that the traffic is harmless. Hence, the **IDS gets more packets** than the destination

- An attacker sends one-character packets to the target system via the IDS with **varying TTL** such that some packets reach the IDS but not the target system
- This will result in the IDS and the target system having **two different character strings**

Insertion Attack

Insertion is the process by which the attacker confuses the IDS by forcing it to read invalid packets (i.e., the system may not accept the packet addressed to it). An IDS blindly trusts and accepts a packet that an end system rejects. If a packet is malformed or if it does not reach its actual destination, the packet is invalid. If the IDS reads an invalid packet, it gets confused. An attacker exploits this condition and inserts data into the IDS. This attack occurs when the NIDS is less strict in processing packets than the internal network. The attacker obscures extra traffic and the IDS concludes that the traffic is harmless. Hence, the IDS gets more packets than the destination.

To understand how insertion becomes a problem for a network IDS, it is important to understand how the IDS detects attacks. It employs pattern-matching algorithms to look for specific patterns of data in a packet or stream of packets. For example, it might search for the "phf" string in an HTTP request to discover a PHF **Common Gateway Interface (CGI)** attack. An attacker who can insert packets into the IDS can prevent pattern matching from working. For instance, an attacker can send the string "phf" to a web server, attempting to exploit the CGI vulnerability, but force the IDS to read "phoneyf" (by "inserting" the string "oney") instead. A straightforward insertion attack involves intentionally corrupting the IP checksum. Every packet transmitted on an IP network has a checksum that verifies the corrupted packets. IP checksums are 16-bit numbers computed by examining the information in the packet. If the checksum on an IP packet does not match the actual packet, the addressed host will not accept it, while the IDS might consider it as part of the effective stream.

For example, the attacker can send packets whose time-to-live (TTL) fields are crafted to reach the IDS but not the target computers. This will result in the IDS and the target system having two different character strings. An attacker confronts the IDS with a stream of one-character packets (the attacker-originated data stream), in which one of the characters (the letter "X") will be accepted only by the IDS. As a result, the IDS and the end system reconstruct two different strings.

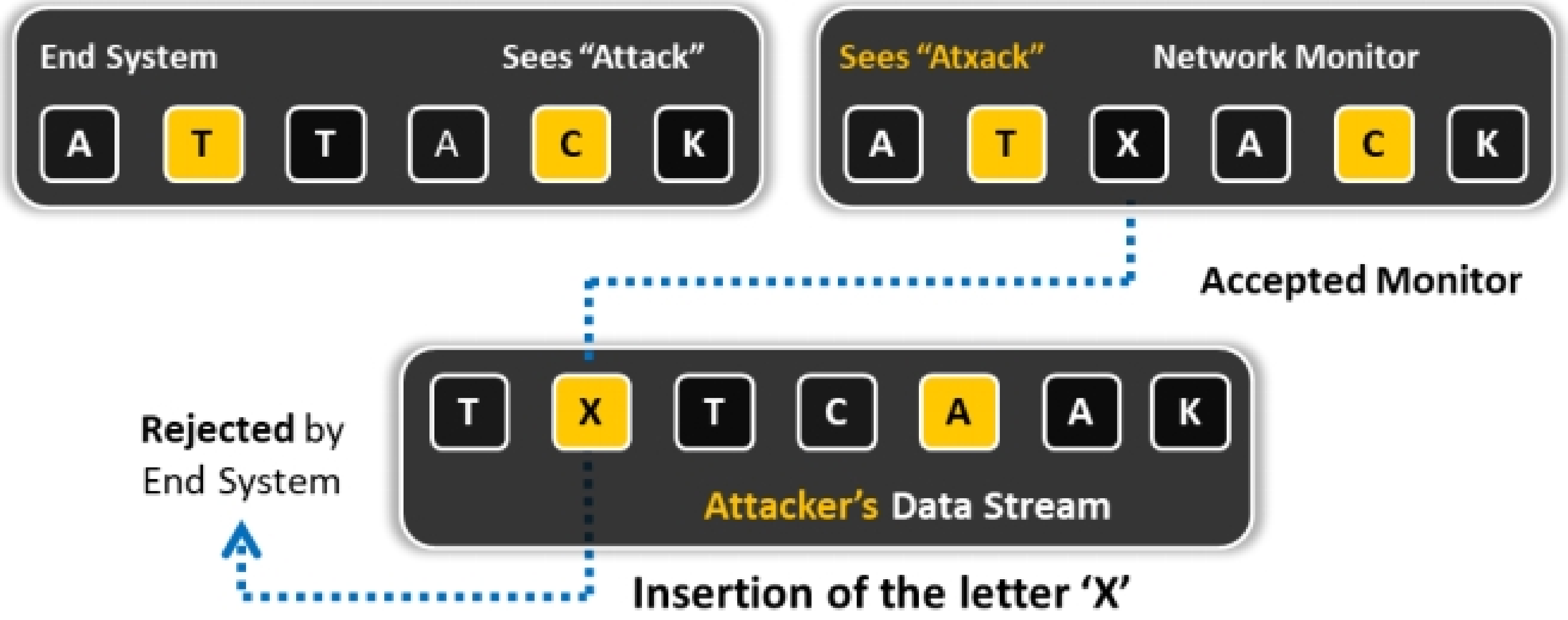
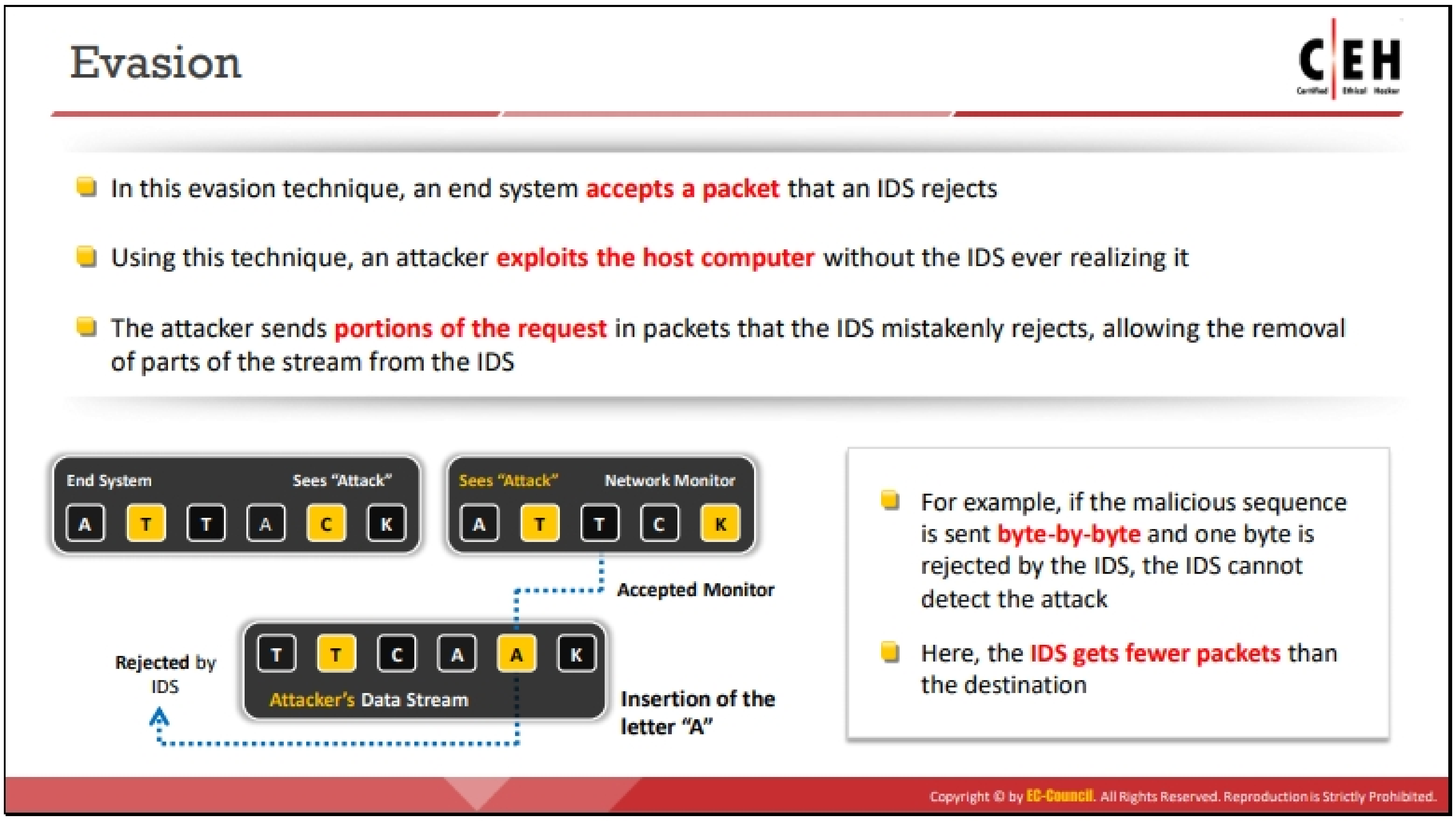

End System
Sees "Attack"
A T T A C K
Sees "Atxack"
Network Monitor
A T X A C K
Accepted Monitor
Rejected by
End System
T X T C A A K
Attacker's Data Stream
Insertion of the letter 'X'

Figure 12.34: Evading IDS using Insertion attack



Evasion

- In this evasion technique, an end system **accepts a packet** that an IDS rejects
- Using this technique, an attacker **exploits the host computer** without the IDS ever realizing it
- The attacker sends **portions of the request** in packets that the IDS mistakenly rejects, allowing the removal of parts of the stream from the IDS

End System
Sees "Attack"
A T T A C K
Sees "Attack"
Network Monitor
A T T C K
Accepted Monitor
Rejected by IDS
T T C A A K
Attacker's Data Stream
Insertion of the letter "A"

- For example, if the malicious sequence is sent **byte-by-byte** and one byte is rejected by the IDS, the IDS cannot detect the attack
- Here, the **IDS gets fewer packets** than the destination

Evasion

An "evasion" attack occurs when the IDS discards packets while the host that has to get the packets accepts them. Using this technique, an attacker exploits the host computer. Evasion attacks have an adverse effect on the accuracy of the IDS. An evasion attack at the IP layer allows an attacker to attempt arbitrary attacks against hosts on a network without the IDS ever realizing it. The attacker sends portions of the request in packets that the IDS mistakenly rejects, allowing the removal of parts of the stream from the ID system's view. For example, if the attacker sends a malicious sequence byte by byte, and if the IDS rejects only one byte, it cannot detect the attack. Here, the IDS gets fewer packets than the destination.

One example of an evasion attack is when an attacker opens a TCP connection with a data packet. Before any TCP connection can be used, it must be "**opened**" with a handshake between the two endpoints of the connection. An essential fact about TCP is that the handshake packets can themselves bear data. The IDS that does not accept the data in these packets is vulnerable to an evasion attack.

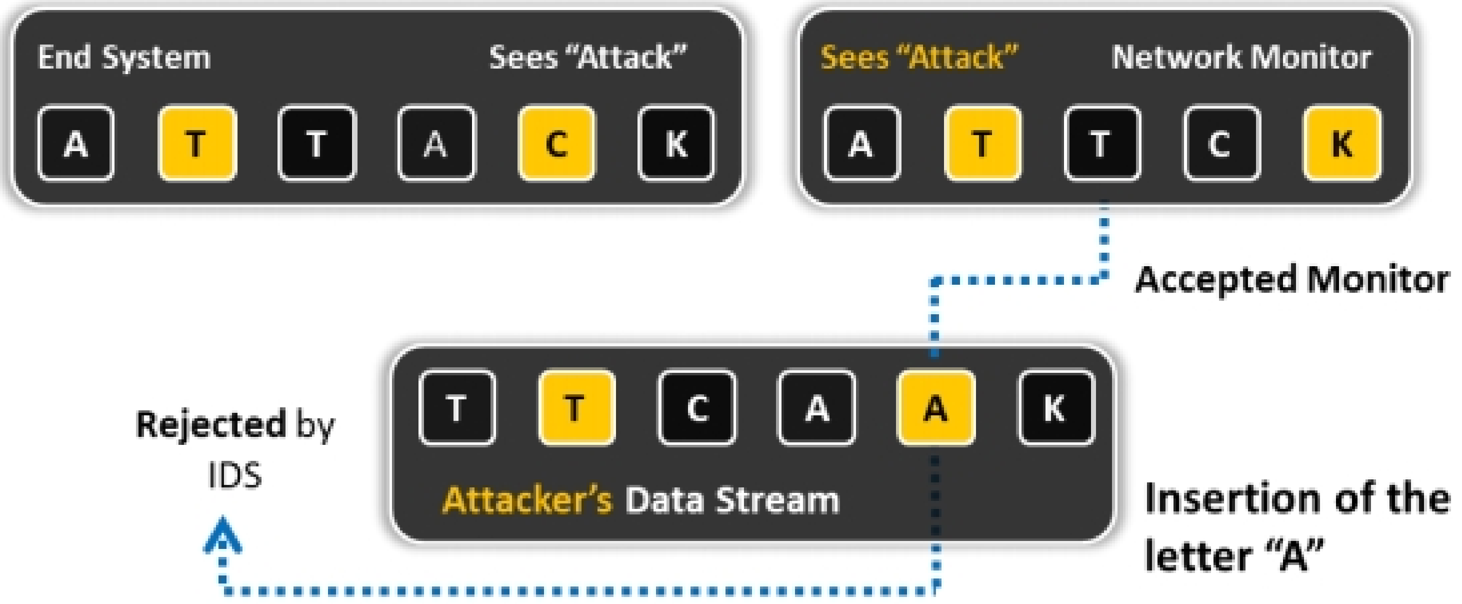
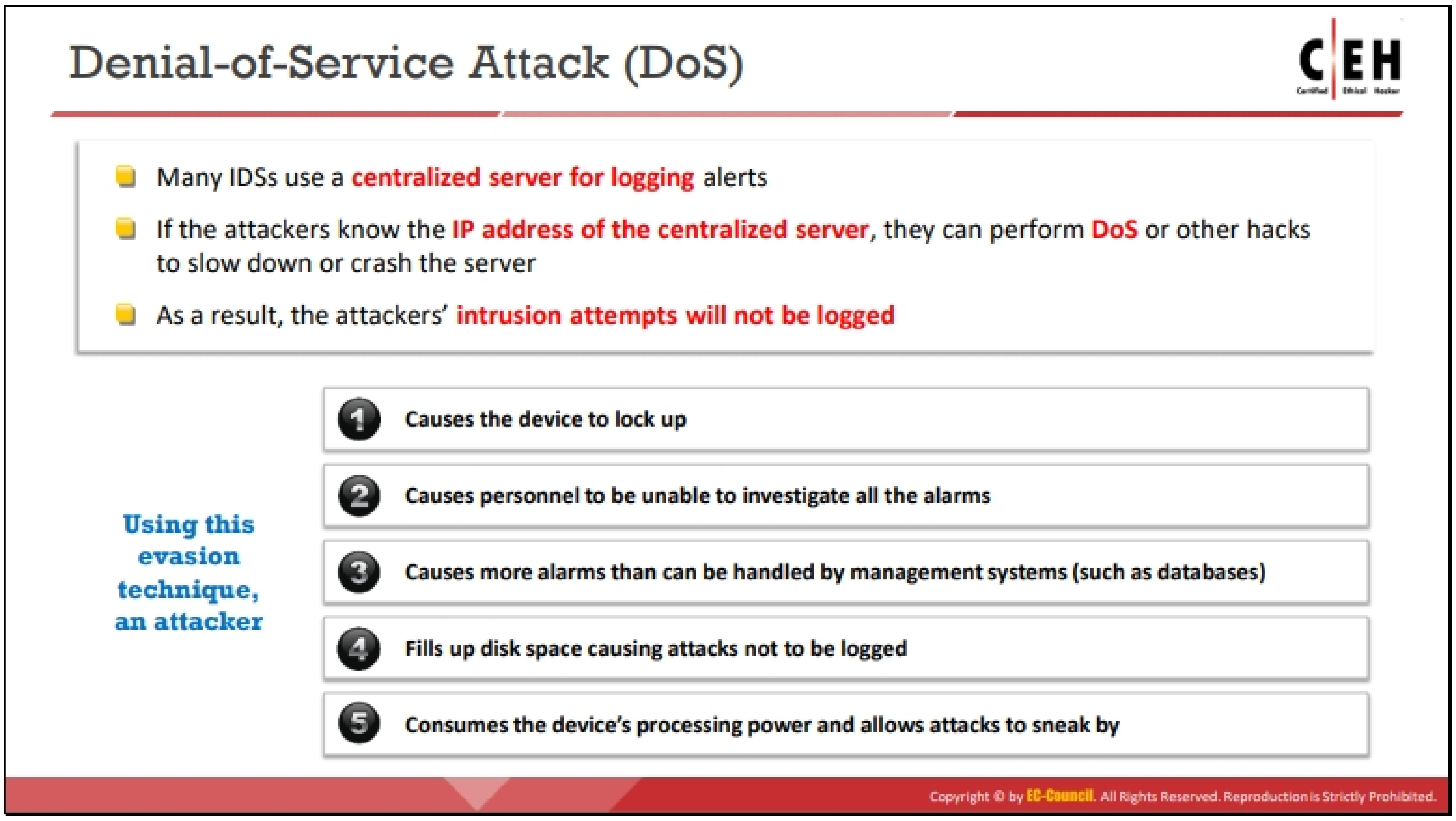

End System
Sees "Attack"
A T T A C K
Sees "Attack"
Network Monitor
A T T C K
Accepted Monitor
Rejected by IDS
T T C A A K
Attacker's Data Stream
Insertion of the letter "A"

Figure 12.35: Illustration of Evasion technique



Denial-of-Service Attack (DoS)

- Many IDSs use a **centralized server for logging** alerts
- If the attackers know the **IP address of the centralized server**, they can perform **DoS** or other hacks to slow down or crash the server
- As a result, the attackers' **intrusion attempts will not be logged**

Using this evasion technique, an attacker

1. Causes the device to lock up
2. Causes personnel to be unable to investigate all the alarms
3. Causes more alarms than can be handled by management systems (such as databases)
4. Fills up disk space causing attacks not to be logged
5. Consumes the device's processing power and allows attacks to sneak by

Denial-of-Service Attack (DoS)

Multiple types of DoS attack will work against IDS. The attacker identifies a point of network processing that requires the allocation of a resource, causing a condition to occur in which all of that resource is consumed. The resources affected by the attacker are CPU cycles, memory, disk space, and network bandwidth. Attackers monitor and attack the CPU capabilities of the IDS. This is because the IDS needs half of a CPU cycle to read the packets, detect the purpose of their existence, and then compare them with some location in the saved network state. An attacker can verify the most computationally expensive network processing operations and then compel the IDS to spend all its time in carrying out useless work.

An IDS requires memory for a variety of tasks such as generating a match for the patterns, saving the TCP connections, maintaining reassembly queues, and producing buffers for the data. In the initial phase, the system requires memory to read the packets. The system will allocate the memory for network processing operations. An attacker can verify the processing operations that require the IDS to allocate memory and force the IDS to assign all of its memory for meaningless information.

In certain circumstances, the IDS store activity logs on the disk. The stored events occupy most of the disk space. Most computers have limited disk space. The attackers can occupy a significant part of the disk space on the IDS by creating and storing a large number of useless events. This renders the IDS useless in terms of storing real events.

Network IDS record the activity on the networks they monitor. They are competent because networks are rarely used to their full capacity; few monitoring systems can cope with an extremely busy network.

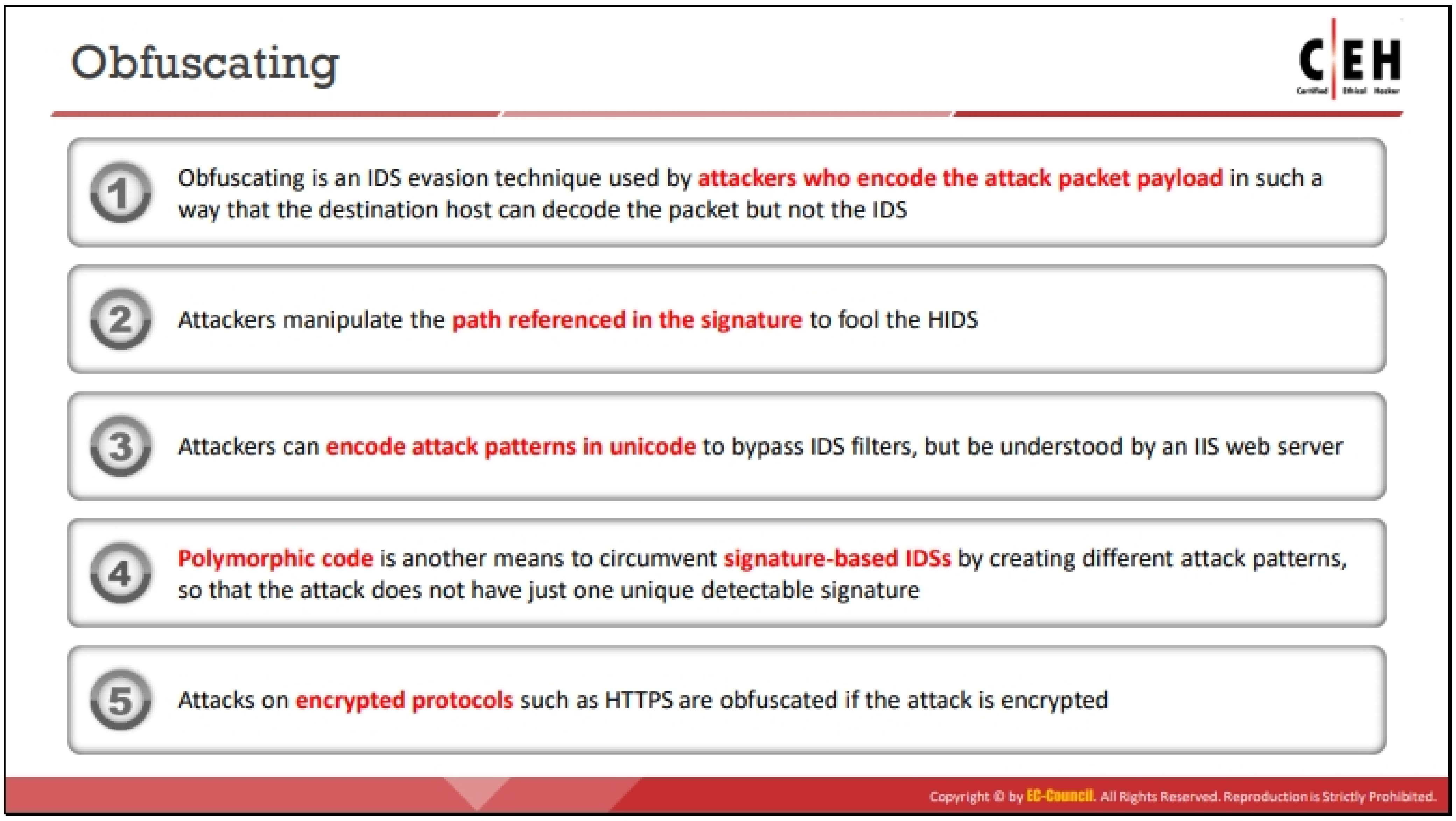
The IDS, unlike an end system, must read everyone's packets, not just those explicitly sent to it. An attacker can overload the network with meaningless information and prevent the IDS from keeping up with what is happening on the network.

Many IDS today employ central logging servers that are used exclusively to store IDS alert logs. The central server's function is to centralize alert data so that it is viewed as a whole rather than on a system-by-system basis.

However, if attackers know the central log server's IP address, they could slow it down or even crash it using a DoS attack. After shutting down the server, attacks could go unnoticed because the alert data is now no longer logged.

Using this evasion technique, an attacker

- Causes the device to lock up
- Causes personnel to be unable to investigate all the alarms
- Causes more alarms than can be handled by management systems (such as databases, etc.)
- Fills up disk space, preventing attacks from being logged
- Consumes the device's processing power and allows attacks to sneak by



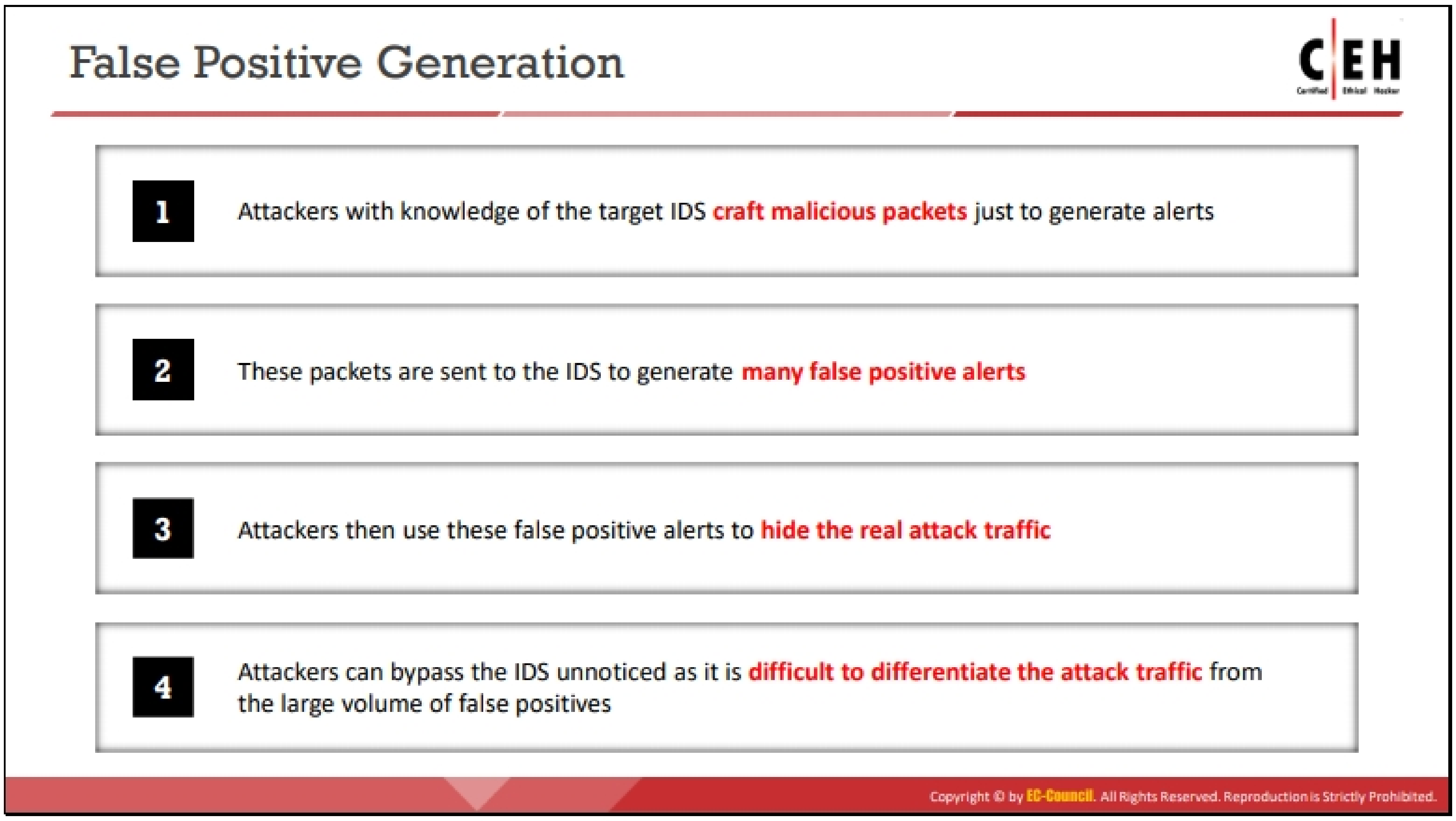
Obfuscating

1. Obfuscating is an IDS evasion technique used by **attackers who encode the attack packet payload** in such a way that the destination host can decode the packet but not the IDS
2. Attackers manipulate the **path referenced in the signature** to fool the HIDS
3. Attackers can **encode attack patterns in unicode** to bypass IDS filters, but be understood by an IIS web server
4. **Polymorphic code** is another means to circumvent **signature-based IDSs** by creating different attack patterns, so that the attack does not have just one unique detectable signature
5. Attacks on **encrypted protocols** such as HTTPS are obfuscated if the attack is encrypted

Obfuscating

Obfuscation means to make code more difficult to understand or read, generally for privacy or security purposes. A tool called an obfuscator converts a straightforward program into one that works in the same way but which is much more difficult to understand.

Obfuscating is an IDS evasion technique used by attackers to encode the attack packet payload in such a way that the destination host can only decode the packet but not the IDS. An attacker manipulates the path referenced in the signature to fool the HIDS. Using Unicode characters, an attacker can encode attack packets that the IDS would not recognize but which an IIS web server can decode. Polymorphic code is another means to circumvent signature-based IDS by creating unique attack patterns so that the attack does not have a single detectable signature. Attackers perform obfuscated attacks on encrypted protocols such as HTTPS. Attackers can also use obfuscation techniques such as digital steganography to bypass IDS and deploy malware on to the target system lying beyond the IDS.

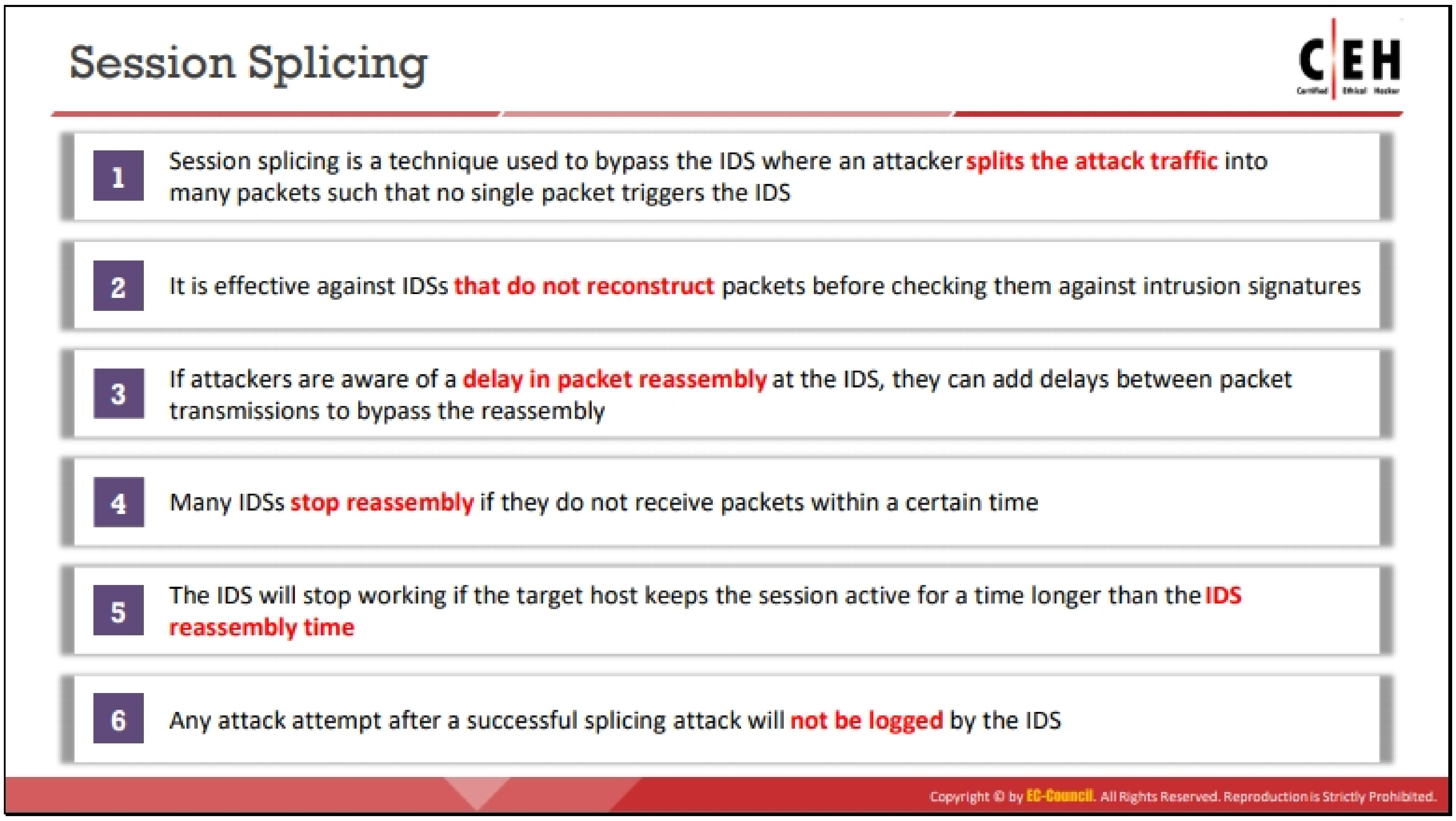


False Positive Generation

1. Attackers with knowledge of the target IDS **craft malicious packets** just to generate alerts
2. These packets are sent to the IDS to generate **many false positive alerts**
3. Attackers then use these false positive alerts to **hide the real attack traffic**
4. Attackers can bypass the IDS unnoticed as it is **difficult to differentiate the attack traffic** from the large volume of false positives

False Positive Generation

This mode does not attack the target; instead, it does something relatively ordinary. In this mode, the IDS generates an alarm when no condition is present to warrant one. Another attack similar to the DoS method is to create a significant amount of alert data that the IDS will log. Attackers construct malicious packets known to trigger alerts within the IDS, forcing it to generate a large number of false reports. Such an attack creates a large amount of log "noise" in an attempt to blend real attacks with fake ones. Attackers know all too well that when looking at log data, it can be challenging to differentiate between legitimate attacks and false positives. If attackers know the IDS, they can even generate false positives specific to that IDS. Attackers then use these false positive alerts to hide real attack traffic. Attackers can bypass IDS unnoticed, as it is difficult to differentiate the attack traffic from the large volume of false positives.



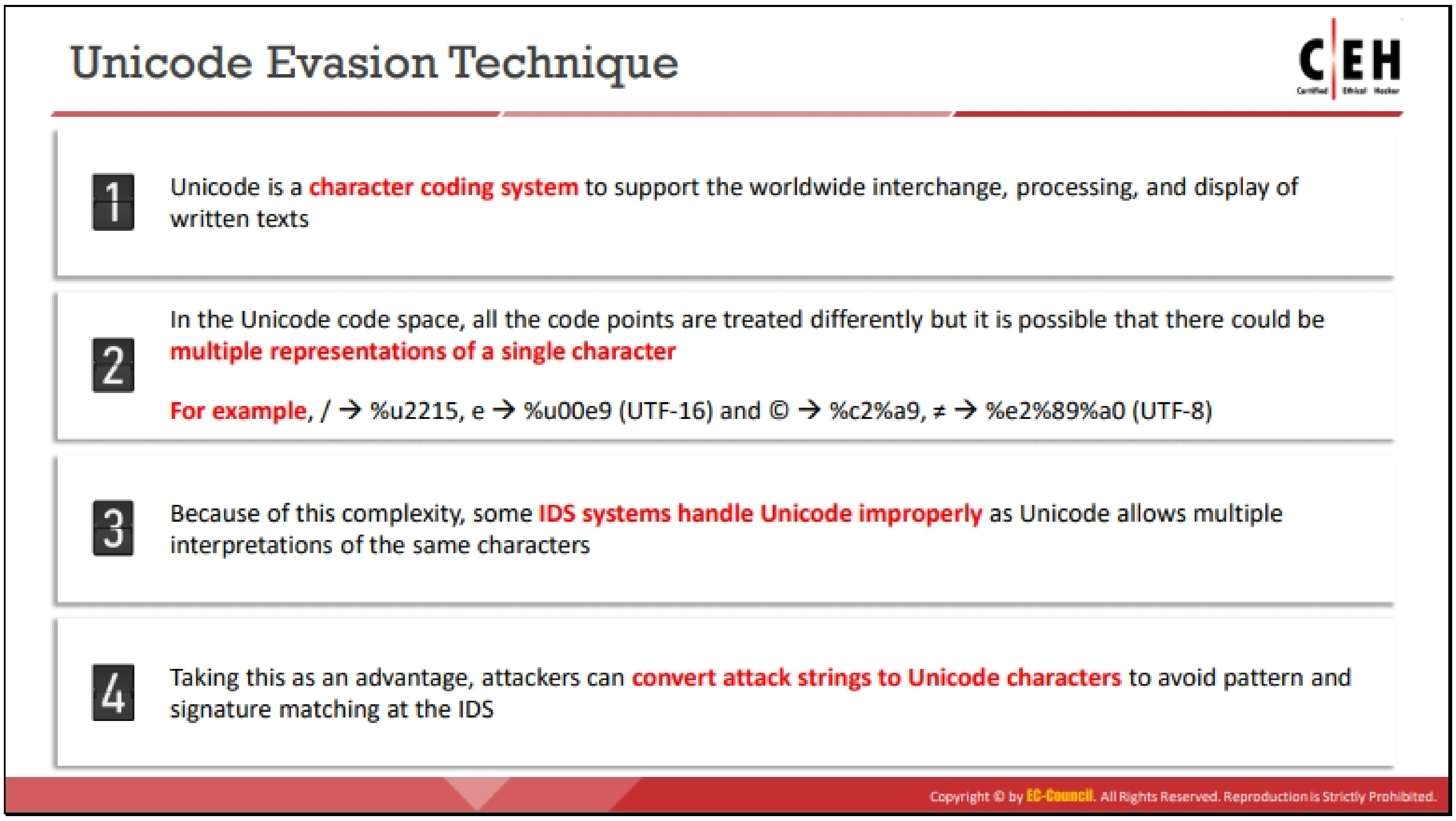
Session Splicing

1. Session splicing is a technique used to bypass the IDS where an attacker **splits the attack traffic** into many packets such that no single packet triggers the IDS
2. It is effective against IDSs **that do not reconstruct** packets before checking them against intrusion signatures
3. If attackers are aware of a **delay in packet reassembly** at the IDS, they can add delays between packet transmissions to bypass the reassembly
4. Many IDSs **stop reassembly** if they do not receive packets within a certain time
5. The IDS will stop working if the target host keeps the session active for a time longer than the **IDS reassembly time**
6. Any attack attempt after a successful splicing attack will **not be logged** by the IDS

Session Splicing

Session splicing is an IDS evasion technique that exploits how some IDS do not reconstruct sessions before pattern-matching the data. It is a network-level evasion method used to bypass IDS where an attacker splits the attack traffic into an excessive number of packets such that no single packet triggers the IDS. The attacker divides the data in the packets into small portions of a few bytes and evades the string match while delivering the data. The IDS cannot handle an excessive number of small-sized packets and fails to detect the attack signatures. If attackers know what IDS is in use, they could add delays between packets to bypass reassembly checking. This approach is effective against IDS that do not reconstruct packets before checking them against intrusion signatures. If attackers are aware of the delay in packet reassembly at the IDS, they can add delays between packet transmissions to bypass the reassembly.

Many IDS reassemble communication streams; hence, if a packet is not received within a reasonable period, many IDS stop reassembling and handling that stream. If the application under attack keeps a session active for a longer time than that spent by the IDS on reassembling it, the IDS will stop. As a result, any session after the IDS stops reassembling the sessions will be susceptible to malicious data theft by attackers. The IDS will not log any attack attempt after a successful splicing attack. Attackers can use tools such as **Nessus** for session-splicing attacks.



Unicode Evasion Technique

1. Unicode is a **character coding system** to support the worldwide interchange, processing, and display of written texts

2. In the Unicode code space, all the code points are treated differently but it is possible that there could be **multiple representations of a single character**

   For example, / → %u2215, e → %u00e9 (UTF-16) and © → %c2%a9, ≠ → %e2%89%a0 (UTF-8)

3. Because of this complexity, some **IDS systems handle Unicode improperly** as Unicode allows multiple interpretations of the same characters

4. Taking this as an advantage, attackers can **convert attack strings to Unicode characters** to avoid pattern and signature matching at the IDS

Unicode Evasion Technique

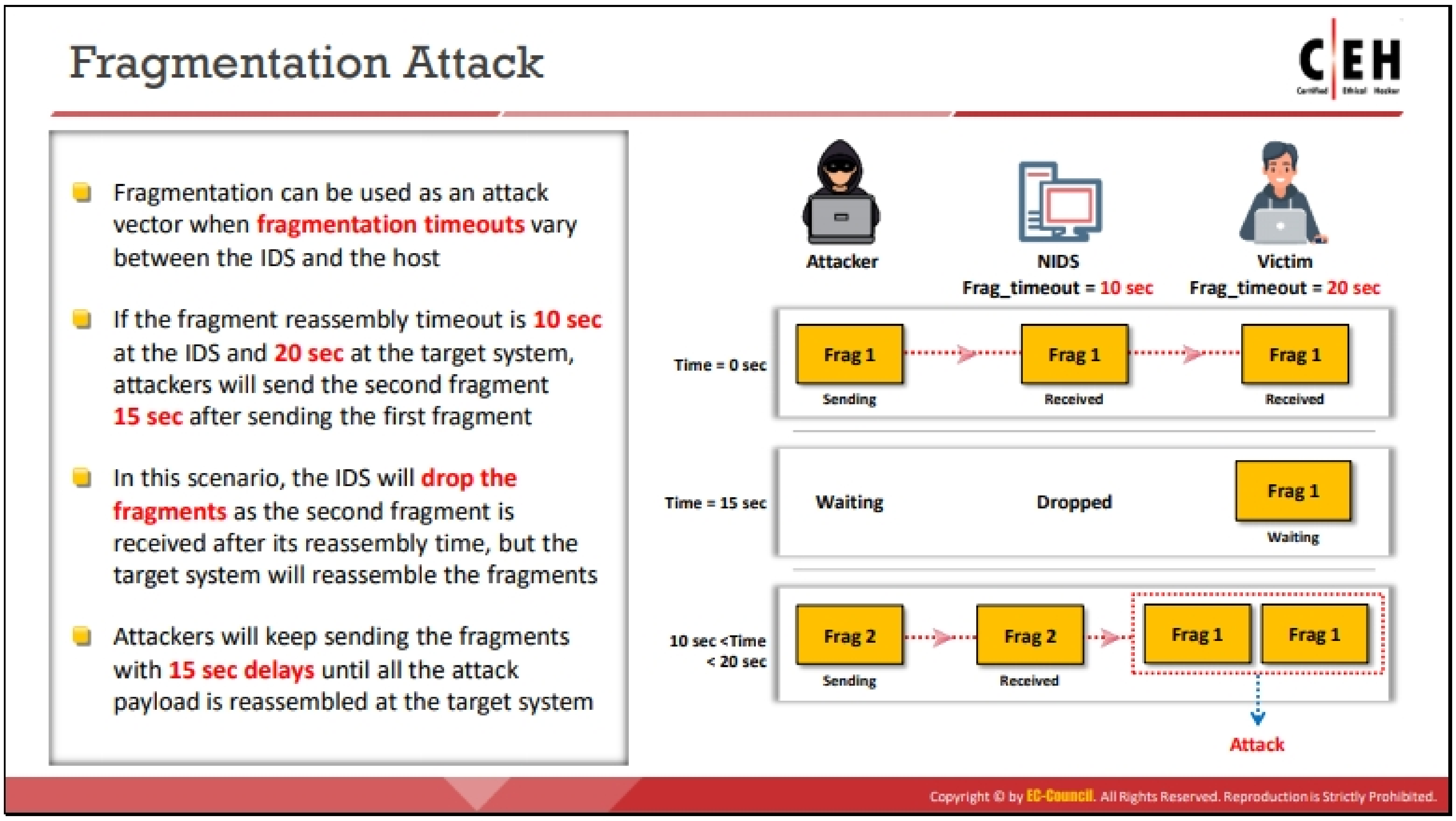
Unicode is a character coding system that supports encoding, processing, and displaying of written texts for universal languages to maintain consistency in a computer representation. Several standards, such as Java, LDAP, and XML, require Unicode, and many OS and applications support it. Attackers can implement an attack by different character encodings known as "**code points**" in the Unicode code space. The most commonly used character encodings are Unicode Transformation Format (UTF)-8 and UTF-16.

For Example: In UTF-16, the character "/" can be represented as "%u2215" and "e" as "%u00e9"; in UTF- 8, "©" can be represented as "%c2%a9" and "≠" as "%e2%89%a0."

Problems with Unicode:

In the Unicode code space, all the code points are treated differently, but it is possible that there are multiple representations of a single character. There are also code points that alter the previous code points. Moreover, applications or OS may assign the same representation to different code points. Because of this complexity, some IDS mishandle Unicode, as Unicode allows multiple interpretations of the same characters.

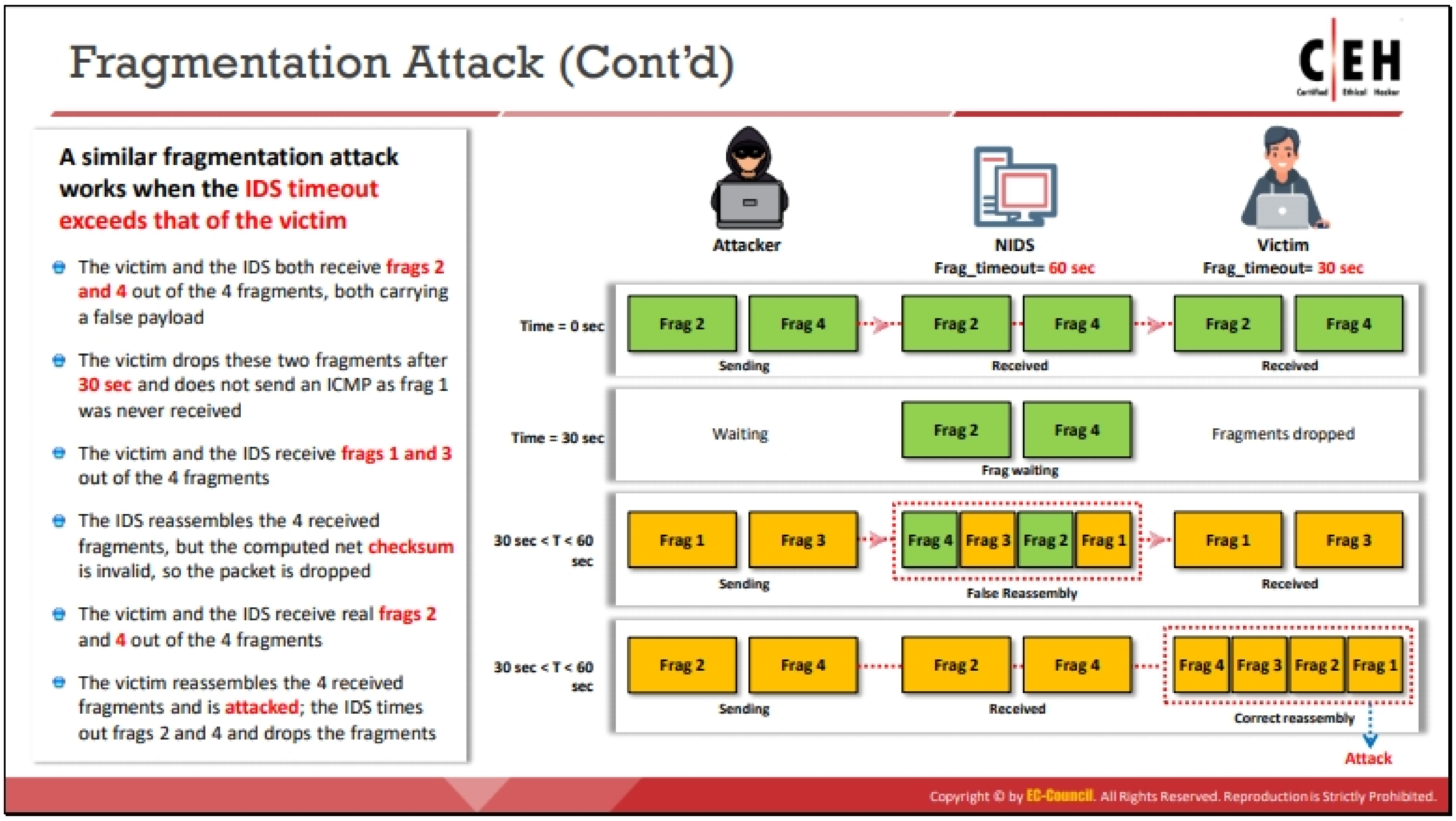
For example, "\" represents 5C, C19C, and E0819C, which makes writing pattern-matching signatures very difficult. Taking advantage of this fact, attackers can convert attack strings into Unicode characters to avoid pattern and signature matching in the IDS. Attackers can also encode URLs in HTTP requests using Unicode characters to bypass HTTP-based attack detection at the IDS.



Fragmentation Attack

- Fragmentation can be used as an attack vector when **fragmentation timeouts** vary between the IDS and the host
- If the fragment reassembly timeout is **10 sec** at the IDS and **20 sec** at the target system, attackers will send the second fragment **15 sec** after sending the first fragment
- In this scenario, the IDS will **drop the fragments** as the second fragment is received after its reassembly time, but the target system will reassemble the fragments
- Attackers will keep sending the fragments with **15 sec delays** until all the attack payload is reassembled at the target system

C|EH
Certified Ethical Hacker
Attacker
NIDS
Frag_timeout = 10 sec
Victim
Frag_timeout = 20 sec
Time = 0 sec
Frag 1
Sending
Frag 1
Received
Frag 1
Received
Time = 15 sec
Waiting
Dropped
Frag 1
Waiting
10 sec <Time < 20 sec
Frag 2
Sending
Frag 2
Received
Frag 1
Frag 1
Attack



Fragmentation Attack (Cont'd)

A similar fragmentation attack works when the IDS timeout exceeds that of the victim

- The victim and the IDS both receive frags 2 and 4 out of the 4 fragments, both carrying a false payload
- The victim drops these two fragments after 30 sec and does not send an ICMP as frag 1 was never received
- The victim and the IDS receive frags 1 and 3 out of the 4 fragments
- The IDS reassembles the 4 received fragments, but the computed net checksum is invalid, so the packet is dropped
- The victim and the IDS receive real frags 2 and 4 out of the 4 fragments
- The victim reassembles the 4 received fragments and is attacked; the IDS times out frags 2 and 4 and drops the fragments

C|EH
Certified Ethical Hacker
Attacker
NIDS
Frag_timeout= 60 sec
Victim
Frag_timeout= 30 sec
Time = 0 sec
Frag 2
Frag 4
Sending
Frag 2
Frag 4
Received
Frag 2
Frag 4
Received
Time = 30 sec
Waiting
Frag 2
Frag 4
Frag waiting
Fragments dropped
30 sec < T < 60 sec
Frag 1
Frag 3
Sending
Frag 4
Frag 3
Frag 2
Frag 1
False Reassembly
Frag 1
Frag 3
Received
30 sec < T < 60 sec
Frag 2
Frag 4
Sending
Frag 2
Frag 4
Received
Frag 4
Frag 3
Frag 2
Frag 1
Correct reassembly
Attack

Fragmentation Attack

IP packets must follow the standard **Maximum Transmission Unit (MTU)** size while traveling across the network. If the packet size is exceeded, it will be split into multiple fragments ("fragmentation"). The IP header contains of a fragment ID, fragment offset, fragment length, fragments flags, and others besides the original data. In a network, the flow of packets is irregular; hence, systems need to keep fragments around, wait for future fragments, and then reassemble them in order. Fragmentation can be used as an attack vector when fragmentation timeouts vary between the IDS and the host. Through the process of fragmenting and reassembling, attackers can send malicious packets over the network to exploit and attack systems. To avoid detection by an IDS, attackers may exploit fragmentation by using the fragment reassembly timeout, which varies from system to system.

- **Attack Scenario - 1**

  If, for example, the fragment reassembly timeout is 10 s at the IDS and 20 s at the target system, attackers will send the second fragment 15 s after sending the first fragment. In this scenario, the IDS will drop the fragment on receiving the second fragment after its reassembly timeout, but the target host will reassemble the fragments. Attackers will continue sending fragments with intervals of 15 s until the attack payload is reassembled at the target system. Thus, the victim will reassemble the fragments and receive the attack code, whereas the IDS will not detect this or generate alerts as the IDS drops the fragments.

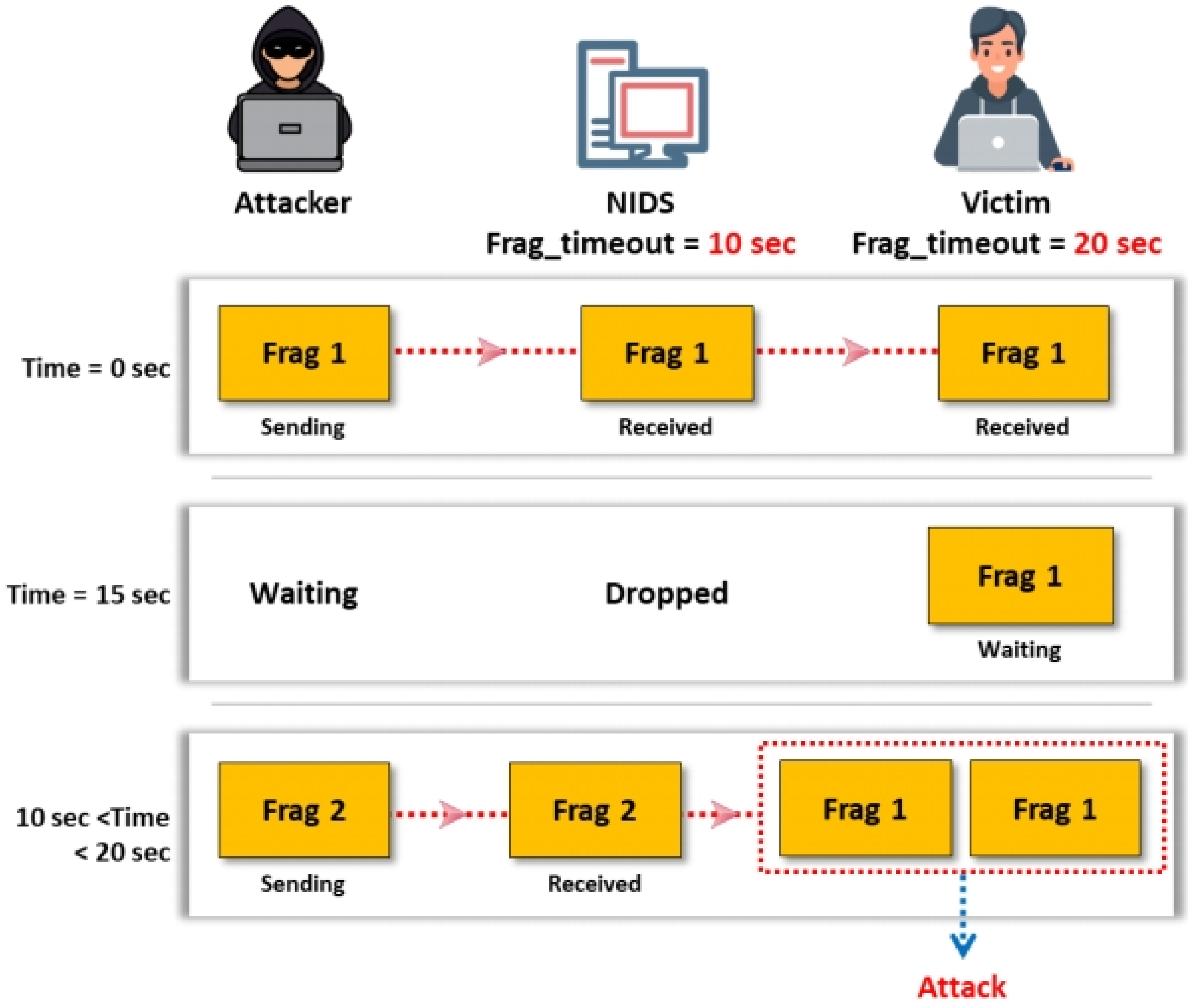

Attacker
NIDS
Frag_timeout = 10 sec
Victim
Frag_timeout = 20 sec
Time = 0 sec
Frag 1
Sending
Frag 1
Received
Frag 1
Received
Time = 15 sec
Waiting
Dropped
Frag 1
Waiting
10 sec <Time < 20 sec
Frag 2
Sending
Frag 2
Received
Frag 1
Frag 1
Attack

Figure 12.36: Fragmentation attack scenario-1

The figure above illustrates the discussed scenario (Attack Scenario-1). The attacker will successfully perform a fragmentation attack on a host. The attacker manipulates the order and time of the fragments and sends those fragments to the victim machine. The attack will succeed when the NIDS fragmentation reassembly timeout is less than the victim's fragmentation reassembly timeout.

- **Attack Scenario - 2**

A similar fragmentation attack works when the IDS timeout exceeds that of the victim. Sometimes, the IDS fragmentation reassembly timeout is greater than that of a host. In this scenario, consider that the attacker has fragmented the attack packet into four fragments: frag-1, frag-2, frag-3, and frag-4. Here, the IDS fragmentation reassembly timeout is 60 s, and the fragmentation reassembly timeout for the host is 30 s.

Initially, the attacker sends frag-2 and frag-4 with a false payload referred to as frag-2' and frag-4', which are received by both the IDS and the victim. The attacker waits until the fragments' reassembly timeout occurs at the victim's system. In this attack, the victim has not received frag-1, so it will drop the fragments without generating an ICMP error message. The attacker then sends a packet (frag-1, frag-3) with a legitimate payload. Now, the victim has only frag-1 and frag-3, whereas the IDS has frag-1, frag-2', frag-3, and frag-4'. Here, frag-2' and frag-4' have false payloads. With the four received fragments, the IDS will perform a TCP reassembly but drop the packet, as the computed checksum for frag-2' and frag-4' will be invalid. If the attacker now sends frag-2 and frag-4 again with a valid payload, the IDS will have only these two fragments with a valid payload, as the previous fragments will have been reassembled and dropped. The victim will have all fragments (frag-1, frag-3, frag-2, frag-4)—with valid payloads that will reassemble—and read the packet as valid.

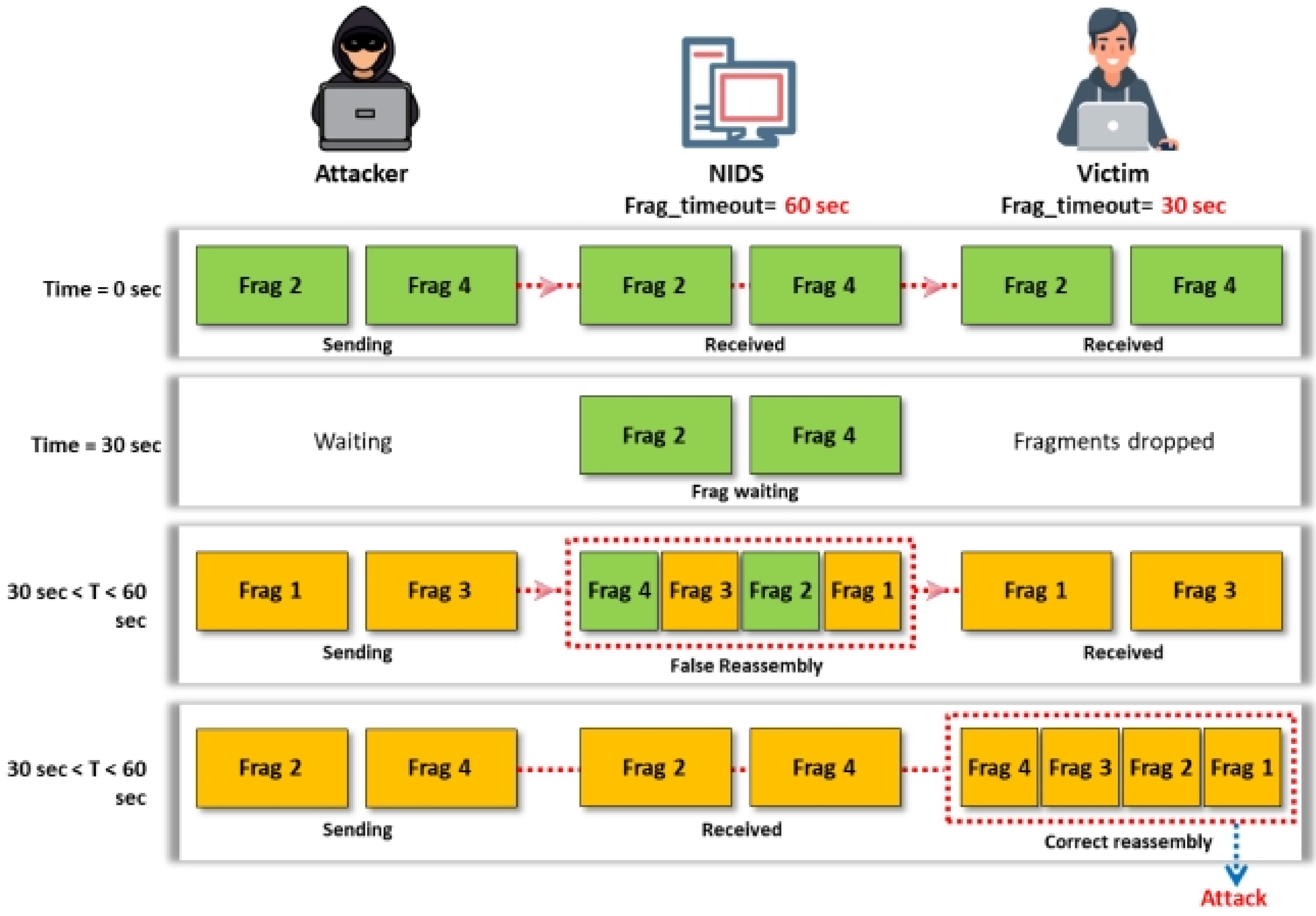
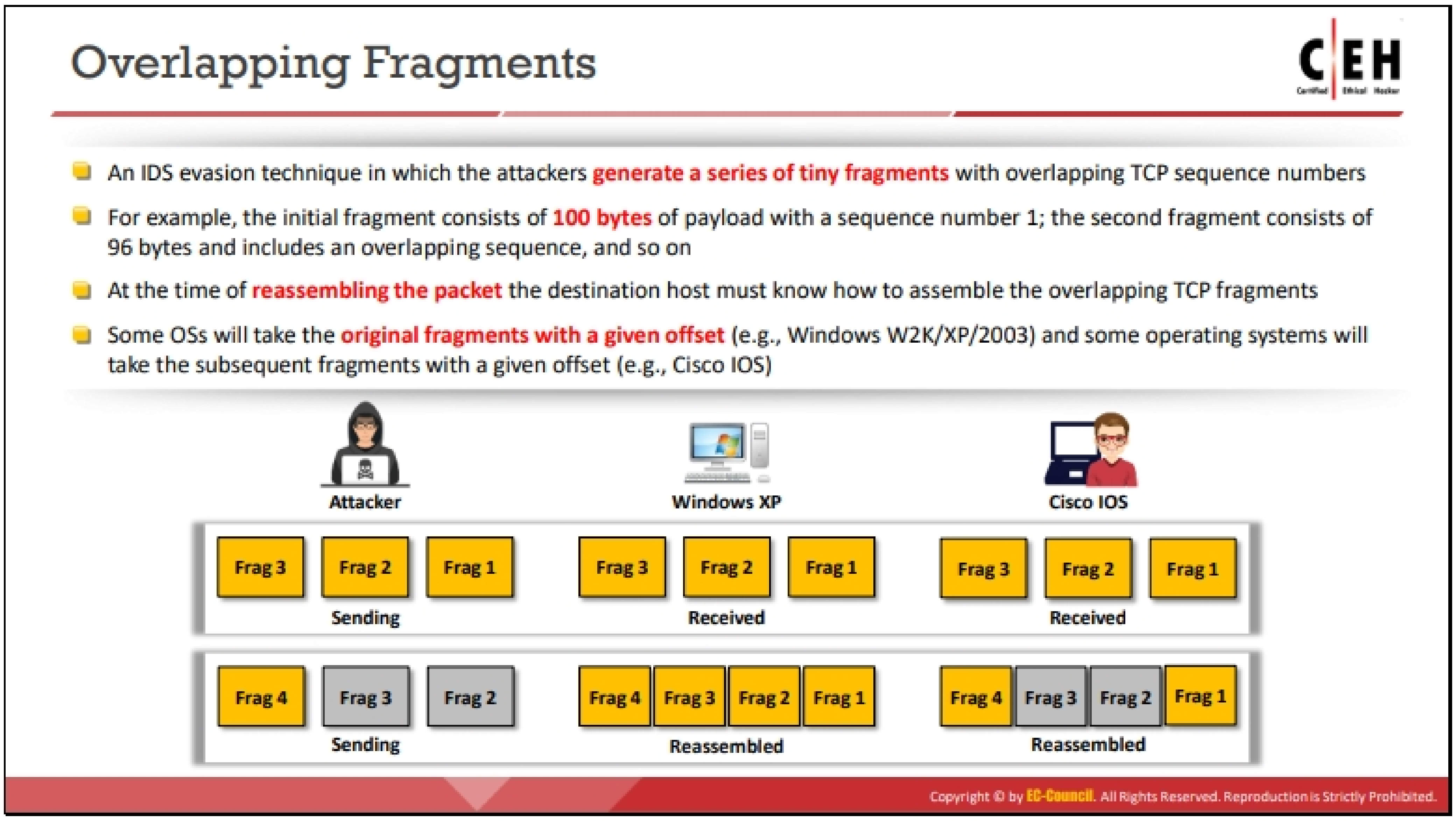

Attacker
NIDS
Frag_timeout= 60 sec
Victim
Frag_timeout= 30 sec
Time = 0 sec
Frag 2
Frag 4
Sending
Received
Received
Time = 30 sec
Waiting
Frag waiting
Fragments dropped
30 sec < T < 60 sec
Frag 1
Frag 3
False Reassembly
30 sec < T < 60 sec
Correct reassembly
Attack

Figure 12.37: Fragmentation attack scenario-2

The figure above illustrates the discussed scenario (Attack Scenario-2). The attacker sends the malicious payload that will falsely reassemble fragments at the IDS and successfully performs a fragmentation attack on a host when the NIDS fragmentation reassembly timeout exceeds the victim's fragmentation reassembly timeout.



Overlapping Fragments

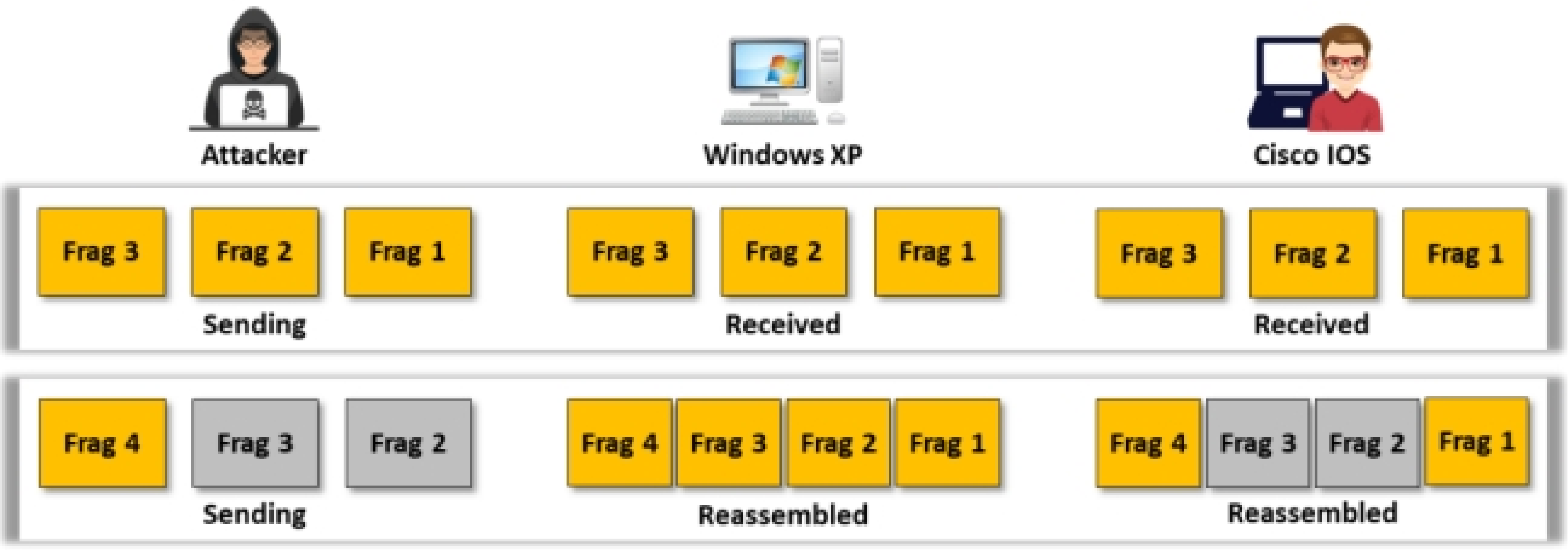
- An IDS evasion technique in which the attackers **generate a series of tiny fragments** with overlapping TCP sequence numbers
- For example, the initial fragment consists of **100 bytes** of payload with a sequence number 1; the second fragment consists of 96 bytes and includes an overlapping sequence, and so on
- At the time of **reassembling the packet** the destination host must know how to assemble the overlapping TCP fragments
- Some OSs will take the **original fragments with a given offset** (e.g., Windows W2K/XP/2003) and some operating systems will take the subsequent fragments with a given offset (e.g., Cisco IOS)

Attacker
Windows XP
Cisco IOS
Frag 3
Frag 2
Frag 1
Sending
Frag 3
Frag 2
Frag 1
Received
Frag 3
Frag 2
Frag 1
Received
Frag 4
Frag 3
Frag 2
Sending
Frag 4
Frag 3
Frag 2
Frag 1
Reassembled
Frag 4
Frag 3
Frag 2
Frag 1
Reassembled

Overlapping Fragments

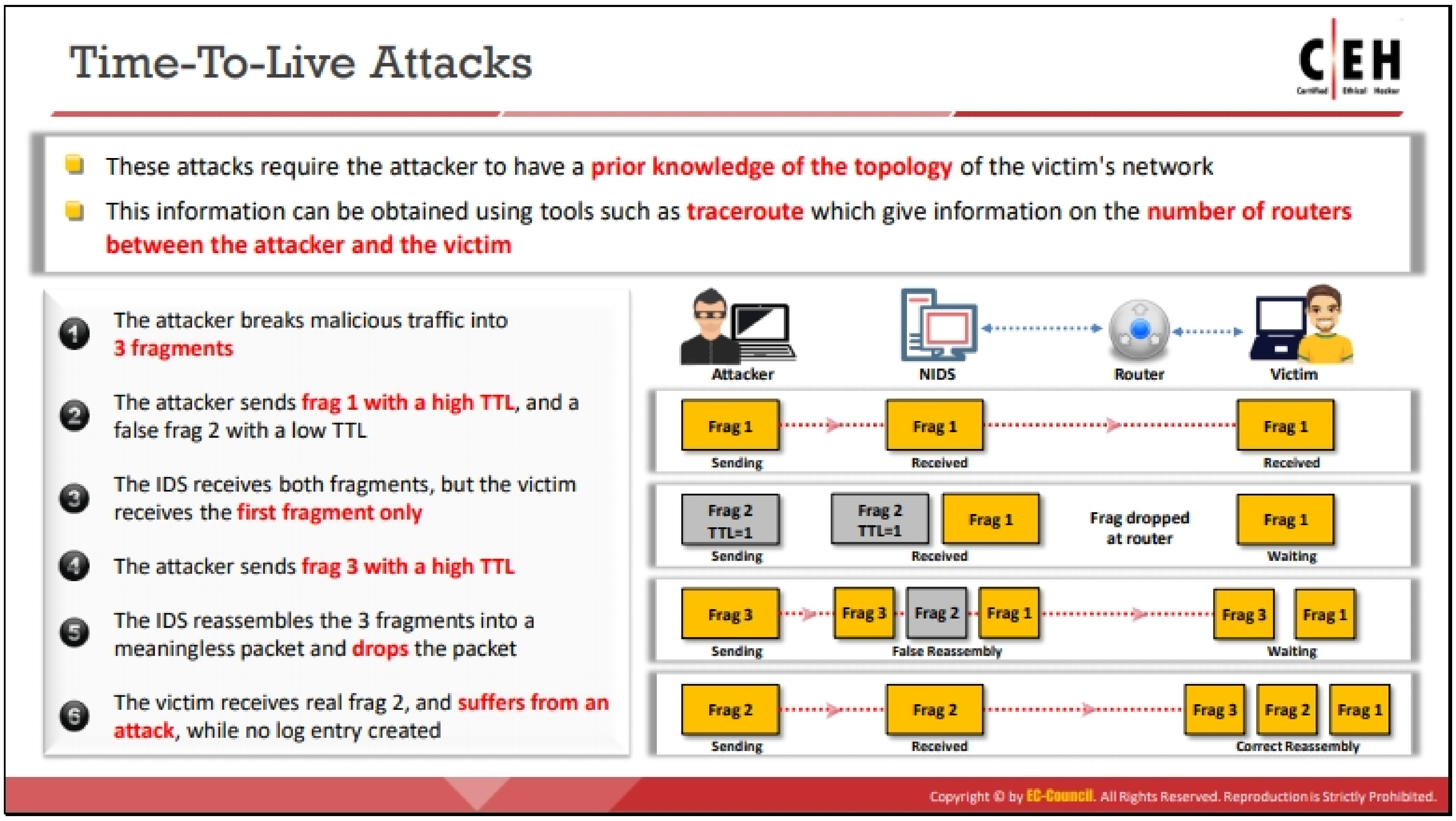
Attackers use overlapping fragments to evade IDS. In this technique, attackers generate a series of tiny fragments with overlapping TCP sequence numbers. For example, the initial fragment consists of 100 bytes of payload with the sequence number of 1, the second fragment includes an overlapping sequence of 96 bytes, and so on. At the time of reassembling the packet, the destination host must know how to assemble the overlapping TCP fragments. Some OS will take the original fragments with a given offset (e.g., Windows W2K/XP/2003) and some OS will take the subsequent fragments with a given offset (e.g., Cisco IOS).

Consider a scenario in which the attacker carries out this attack by breaking the packet into four fragments, sending frag-1, frag-2, and frag-3 first, accepted by both OSs. Then, the attacker sends frag-2', frag-3', and frag-4. Here, the payloads of frag-2' and frag-3' are different from those of frag-2 and frag-3, respectively, but the fragment offset and its length, along with the other fields in the IP header, remain the same. In such a scenario, an OS such as Windows XP will reassemble frag-1, frag-2, frag-3, and frag-4, whereas an OS such as Cisco IOS will reassemble frag-1, frag-2', frag-3', and frag-4.



Attacker
Windows XP
Cisco IOS
Frag 3
Frag 2
Frag 1
Sending
Frag 3
Frag 2
Frag 1
Received
Frag 3
Frag 2
Frag 1
Received
Frag 4
Frag 3
Frag 2
Sending
Frag 4
Frag 3
Frag 2
Frag 1
Reassembled
Frag 4
Frag 3
Frag 2
Frag 1
Reassembled

Figure 12.38: Evading IDS using Overlapping Fragments



Time-To-Live Attacks

- These attacks require the attacker to have a **prior knowledge of the topology** of the victim's network
- This information can be obtained using tools such as **traceroute** which give information on the **number of routers between the attacker and the victim**

1. The attacker breaks malicious traffic into **3 fragments**
2. The attacker sends **frag 1 with a high TTL**, and a false frag 2 with a low TTL
3. The IDS receives both fragments, but the victim receives the **first fragment only**
4. The attacker sends **frag 3 with a high TTL**
5. The IDS reassembles the 3 fragments into a meaningless packet and **drops** the packet
6. The victim receives real frag 2, and **suffers from an attack**, while no log entry created

Time-To-Live Attacks

Each IP packet has a field called **Time to Live (TTL)**, which indicates how many hops the packet can take before a network node discards it. Each router along a data path decrements this value by 1. When the TTL reaches 0, the packet is dropped, and an ICMP alert notification is sent to the sender. Typically, when a host sends a packet, it sets the TTL to a high value such that it can reach its destination under normal circumstances. Different OS use different default initial values for the TTL. Therefore, attackers can guess the number of routers between them and a sending machine, and make assumptions as to what the initial TTL was, thereby guessing which OS a host is running, as a prelude to an attack. To prevent such detection, **SmartDefense** can change the TTL field of all packets (or all outgoing packets) to a given number. These attacks require the attacker to have prior knowledge of the topology of the victim's network. This information can be obtained using tools such as traceroute, which gives information on the number of routers between the attacker and the victim.

Consider a scenario in which a router is present between the IDS and a victim. Attackers need to acquire this information before launching the TTL attack by breaking the malicious data packet into three fragments. It is assumed that the attacker has prior knowledge about the topology of the target network (i.e., how many routers are there between the attacker and victim machines). The attacker fragments the packet and sends frag 1 with the TTL set to a higher value. It is then received by the victim and the IDS. Then, the attacker sends frag-2' with a false payload and a TTL value of 1, which is received by the IDS; however, the victim will not receive it, because the router discards it and the TTL value is reduced to 0. Next, the attacker sends frag-3 with a correct payload and a higher TTL value, which enables it to reach the IDS and the victim. After receiving frag-3, the IDS performs a TCP reassembly on fragments 1, 2', and 3, and the victim waits for frag-2. Finally, the attacker sends frag-2 with a valid payload. The victim, after receiving frag-2, reassembles fragments 1, 2, and 3 and gets the attack code embedded in a malicious payload. Here, the IDS has only frag-2, as it has already reassembled the fragments and the stream has cleared.

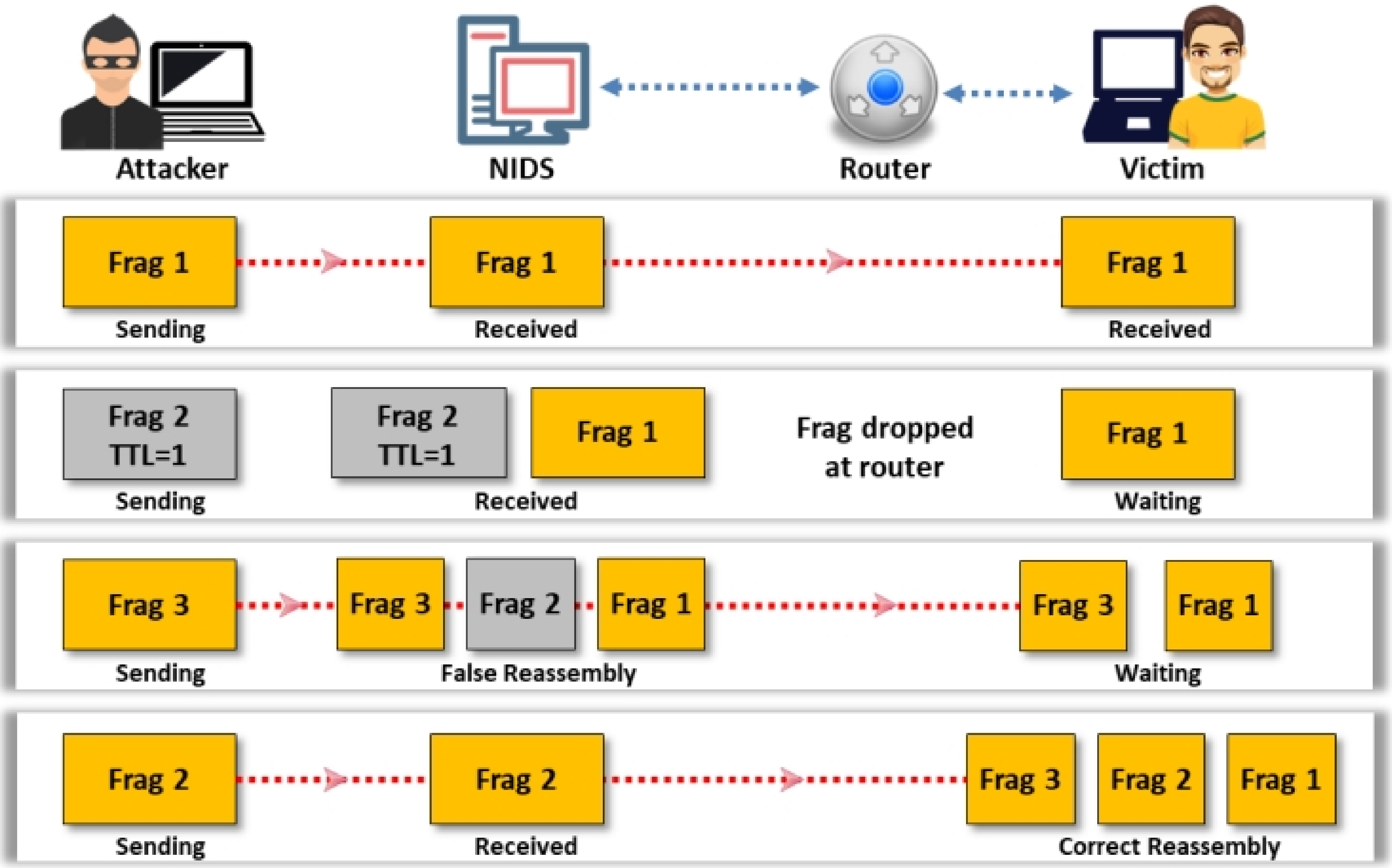
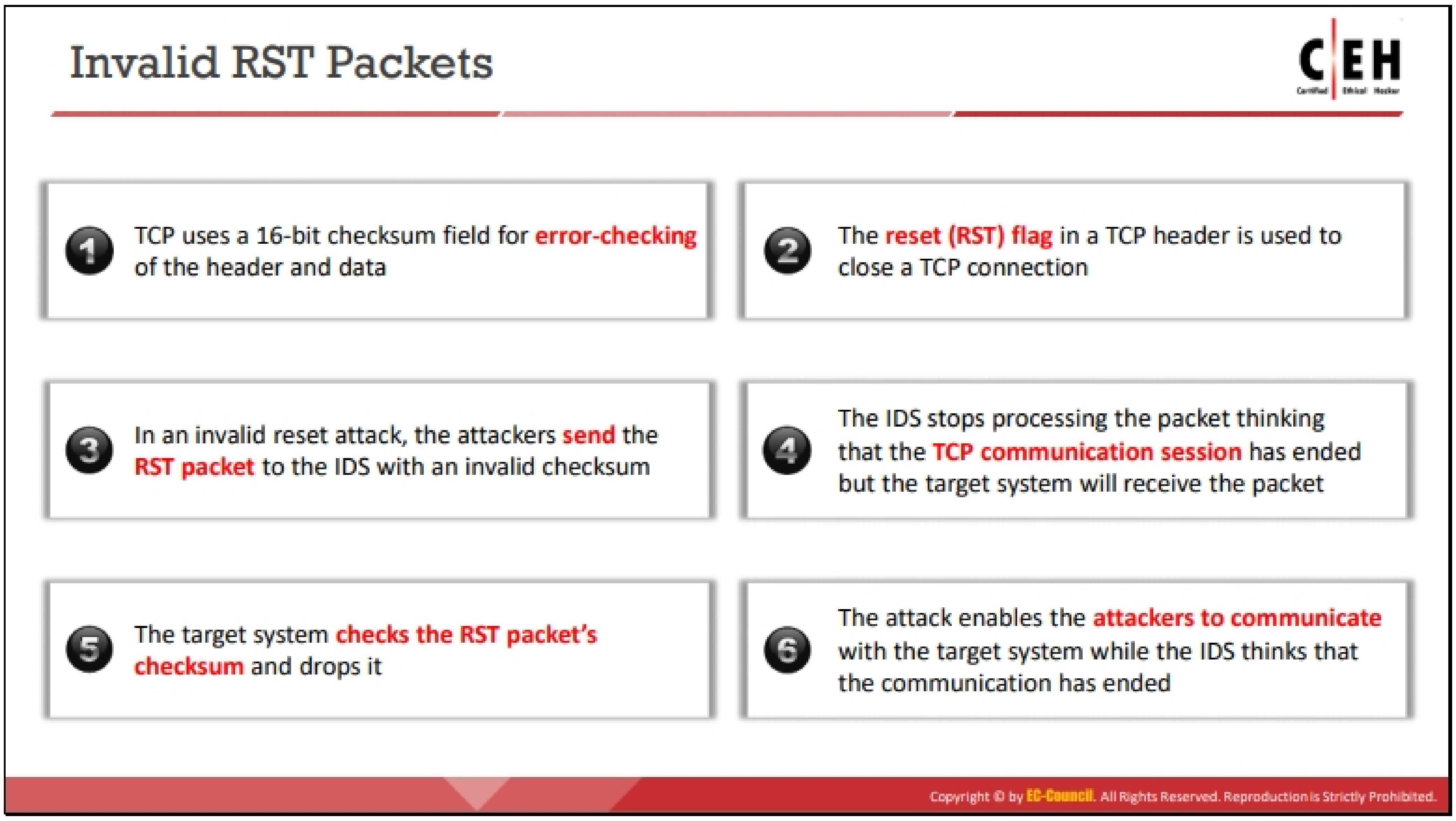

Attacker
NIDS
Router
Victim
Frag 1
Sending
Frag 1
Received
Frag 1
Received
Frag 2 TTL=1
Sending
Frag 2 TTL=1
Frag 1
Received
Frag dropped at router
Frag 1
Waiting
Frag 3
Sending
Frag 3
Frag 2
Frag 1
False Reassembly
Frag 3
Frag 1
Waiting
Frag 2
Sending
Frag 2
Received
Frag 3
Frag 2
Frag 1
Correct Reassembly

Figure 12.39: Evading IDS using Time-To-Live attack



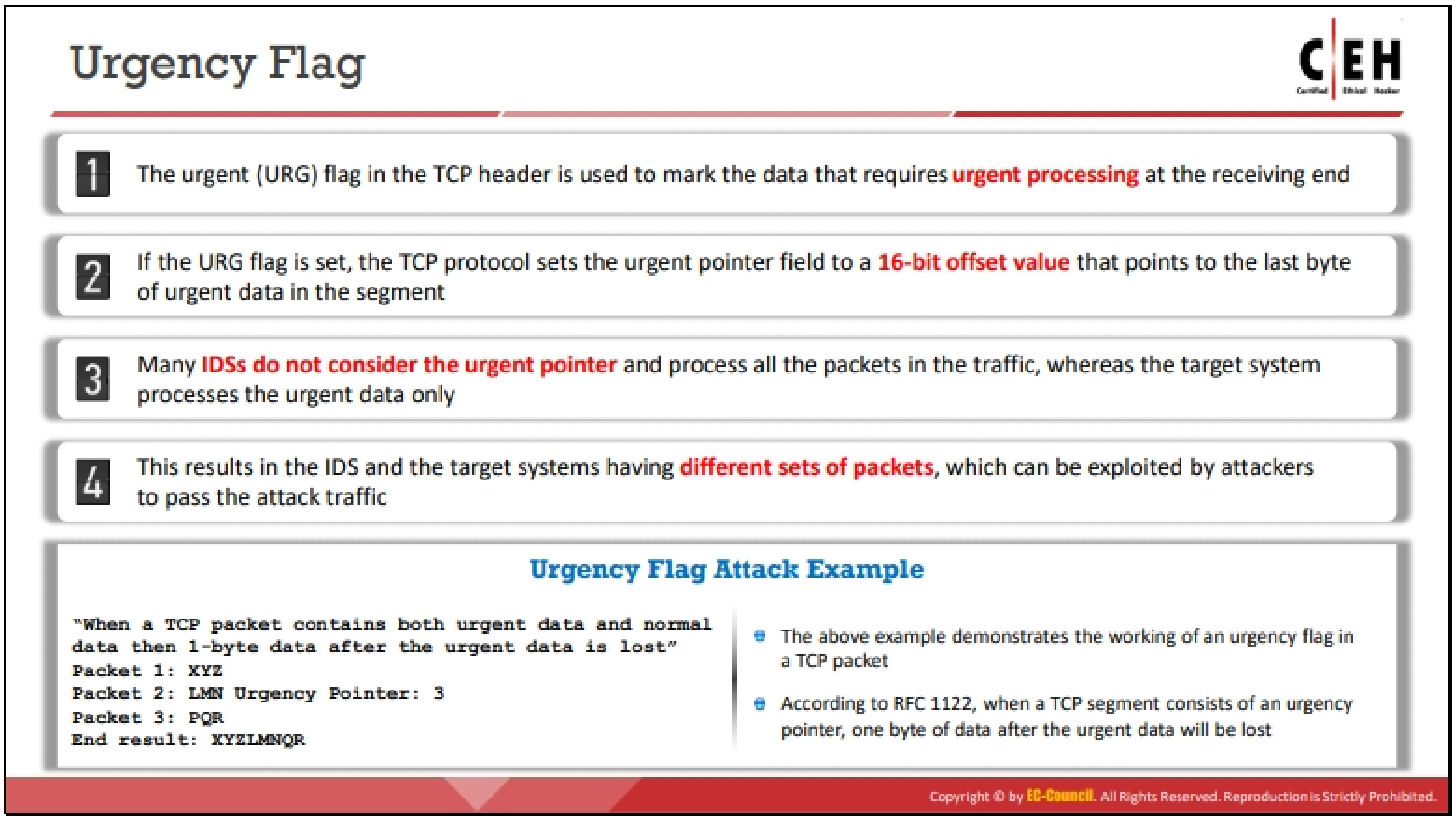
Invalid RST Packets

1. TCP uses a 16-bit checksum field for **error-checking** of the header and data
2. The **reset (RST) flag** in a TCP header is used to close a TCP connection
3. In an invalid reset attack, the attackers **send** the **RST packet** to the IDS with an invalid checksum
4. The IDS stops processing the packet thinking that the **TCP communication session** has ended but the target system will receive the packet
5. The target system **checks the RST packet's checksum** and drops it
6. The attack enables the **attackers to communicate** with the target system while the IDS thinks that the communication has ended

Invalid RST Packets

The TCP uses 16-bit checksums for error checking of the header and data and to ensure that communication is reliable. It adds a checksum to every transmitted segment that is checked at the receiving end. When a checksum differs from the checksum expected by the receiving host, the TCP drops the packet at the receiver's end. The TCP also uses an RST packet to end two-way communications. Attackers can use this feature to elude detection by sending RST packets with an invalid checksum, which causes the IDS to stop processing the stream because the IDS thinks that the communication session has ended. However, the end host checks this packet, verifies the checksum value, and then drops the packet if it is invalid.

Some IDS might interpret this packet as an actual termination of the communication and stop reassembling the communication. Such instances allow attackers to continue to communicate with the end host while confusing the IDS because the end host accepts the packets that follow the RST packet with an invalid checksum value.



Urgency Flag

1. The urgent (URG) flag in the TCP header is used to mark the data that requires **urgent processing** at the receiving end
2. If the URG flag is set, the TCP protocol sets the urgent pointer field to a **16-bit offset value** that points to the last byte of urgent data in the segment
3. Many **IDSs do not consider the urgent pointer** and process all the packets in the traffic, whereas the target system processes the urgent data only
4. This results in the IDS and the target systems having **different sets of packets**, which can be exploited by attackers to pass the attack traffic

Urgency Flag Attack Example

```
"When a TCP packet contains both urgent data and normal
data then 1-byte data after the urgent data is lost"
Packet 1: XYZ
Packet 2: LMN Urgency Pointer: 3
Packet 3: PQR
End result: XYZLMNQR
```

- The above example demonstrates the working of an urgency flag in a TCP packet
- According to RFC 1122, when a TCP segment consists of an urgency pointer, one byte of data after the urgent data will be lost

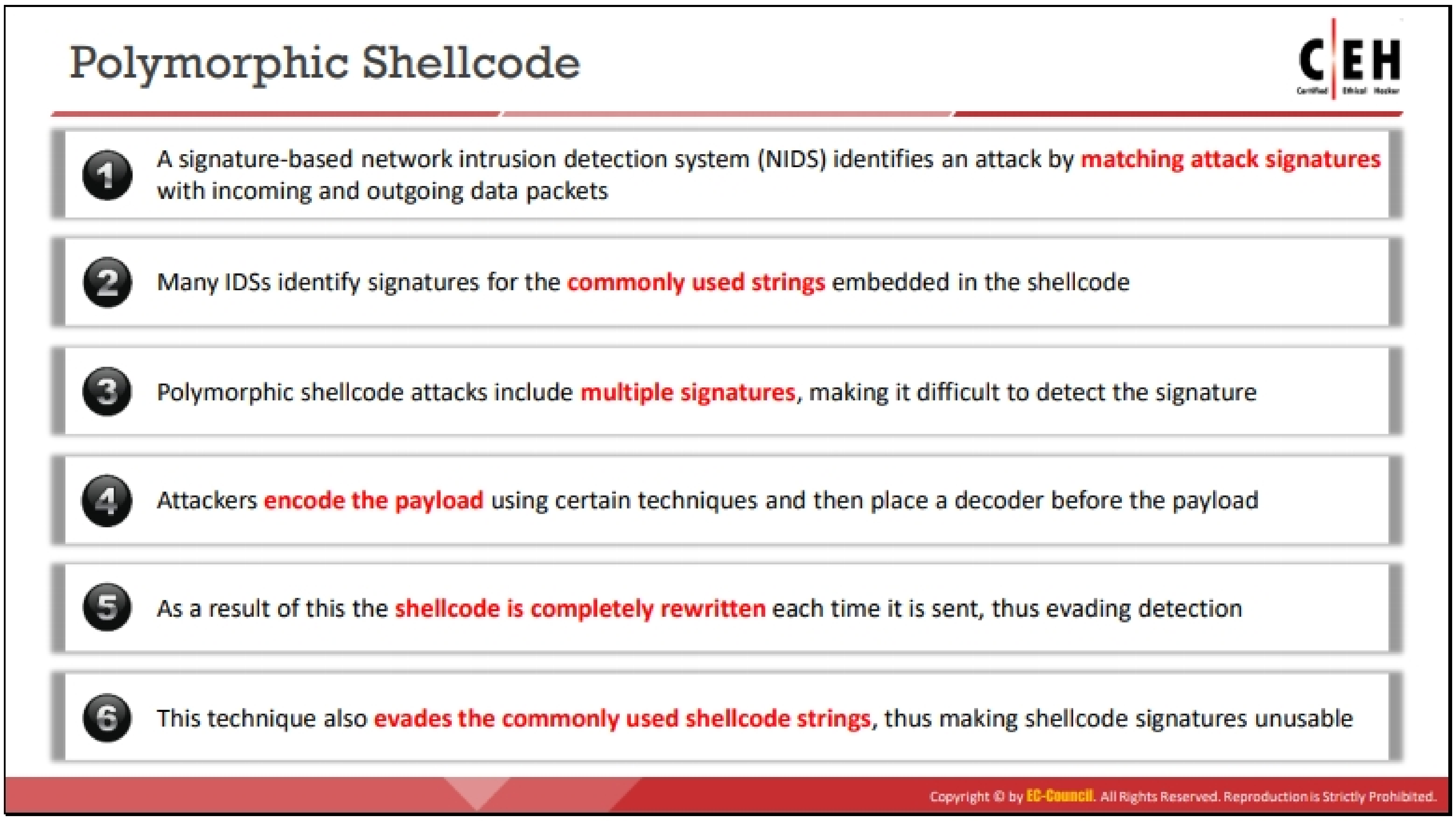
Urgency Flag

The urgency flag in the TCP marks data as urgent. TCP uses an urgency pointer that points to the beginning of urgent data within a packet. When the user sets the urgency flag, the TCP ignores all data before the urgency pointer, and the data to which the urgency pointer points is processed. If the URG flag is set, the TCP sets the Urgent Pointer field to a 16-bit offset value that points to the last byte of urgent data in the segment. Some IDS do not consider the TCP's urgency feature and process all the packets in the traffic, whereas the target system processes only the urgent data. Attackers exploit this feature to evade the IDS, as seen in other evasion techniques. Attackers can place garbage data before the urgency data. The pointer and the IDS read that data without consideration of the end host's urgency flag handling. This means that the IDS have more data than the end host processes. This results in the IDS and the target systems having different sets of packets, which can be exploited by attackers to pass the attack traffic.

Example:

```
"When a TCP packet contains both urgent data and normal data then 1-byte data after the urgent data is lost"

Packet 1: XYZ

Packet 2: LMN Urgency Pointer: 3

Packet 3: PQR

End result: XYZLMNQR
```

The above example demonstrates the working of an urgency flag in a TCP packet. According to RFC 1122, if a TCP segment consists of an urgency pointer, then one byte of data after the urgent data will be lost.



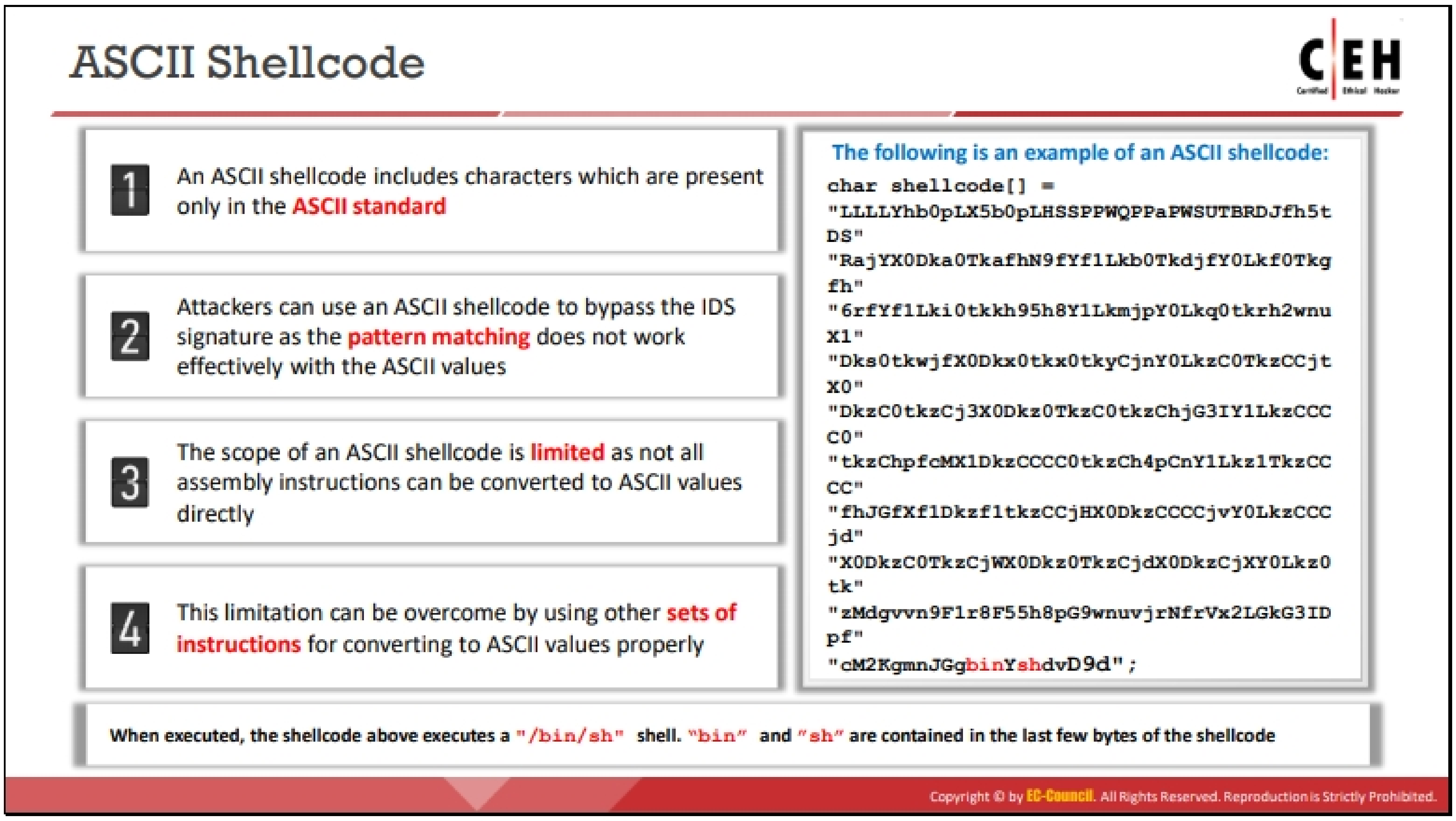
Polymorphic Shellcode

1. A signature-based network intrusion detection system (NIDS) identifies an attack by **matching attack signatures** with incoming and outgoing data packets
2. Many IDSs identify signatures for the **commonly used strings** embedded in the shellcode
3. Polymorphic shellcode attacks include **multiple signatures**, making it difficult to detect the signature
4. Attackers **encode the payload** using certain techniques and then place a decoder before the payload
5. As a result of this the **shellcode is completely rewritten** each time it is sent, thus evading detection
6. This technique also **evades the commonly used shellcode strings**, thus making shellcode signatures unusable

Polymorphic Shellcode

A signature-based network intrusion detection system (NIDS) identifies an attack by matching attack signatures with incoming and outgoing data packets. Many IDS identify signatures for commonly used strings embedded in the shellcode. Polymorphic shellcode attacks include multiple signatures, making it difficult to detect the signature. Attackers encode the payload using some technique and then place a decoder before the payload. As a result, the shellcode is completely rewritten each time it is sent, thereby evading detection.

With polymorphic shellcodes, attackers hide their shellcode (attack code) by encrypting it with an unknown encryption algorithm and including the decryption code as part of the attack packet. To carry out polymorphic shellcode attacks, they use an existing buffer-overflow exploit and set the “return” memory address on the overflowed stack to the entrance point of the decryption code. This makes it difficult for the IDS to identify it as a shellcode. Therefore, when attackers modify/transform their attacks in this way, the NIDS cannot recognize them. This technique also evades commonly used shellcode strings, thus making shellcode signatures unusable.



ASCII Shellcode

1. An ASCII shellcode includes characters which are present only in the **ASCII standard**
2. Attackers can use an ASCII shellcode to bypass the IDS signature as the **pattern matching** does not work effectively with the ASCII values
3. The scope of an ASCII shellcode is **limited** as not all assembly instructions can be converted to ASCII values directly
4. This limitation can be overcome by using other **sets of instructions** for converting to ASCII values properly

The following is an example of an ASCII shellcode:

```
char shellcode[] =
"LLLLYhb0pLX5b0pLHSSPPWQPPaPWSUTBRDJfh5t
DS"
"RajYX0Dka0TkafhN9fYf1Lkb0TkdjfY0Lkf0Tkg
fh"
"6rfYf1Lki0tkkh95h8Y1LkmjpY0Lkq0tkrh2wnu
X1"
"Dks0tkwjfX0Dkx0tkx0tkyCjnY0LkzC0TkzCCjt
X0"
"DkzC0tkzCj3X0Dkz0TkzC0tkzChjG3IY1LkzCCC
C0"
"tkzChpfcMX1DkzCCCC0tkzCh4pCnY1Lkz1TkzCC
CC"
"fhJGfXf1Dkzf1tkzCCjHX0DkzCCCCjvY0LkzCCC
jd"
"X0DkzC0TkzCjWX0Dkz0TkzCjdX0DkzCjXY0Lkz0
tk"
"zMdgvvn9F1r8F55h8pG9wnuvjrNfrVx2LGkG3ID
pf"
"cM2KgmnJGgbinYshdvD9d";
```

When executed, the shellcode above executes a "/bin/sh" shell. "bin" and "sh" are contained in the last few bytes of the shellcode

ASCII Shellcode

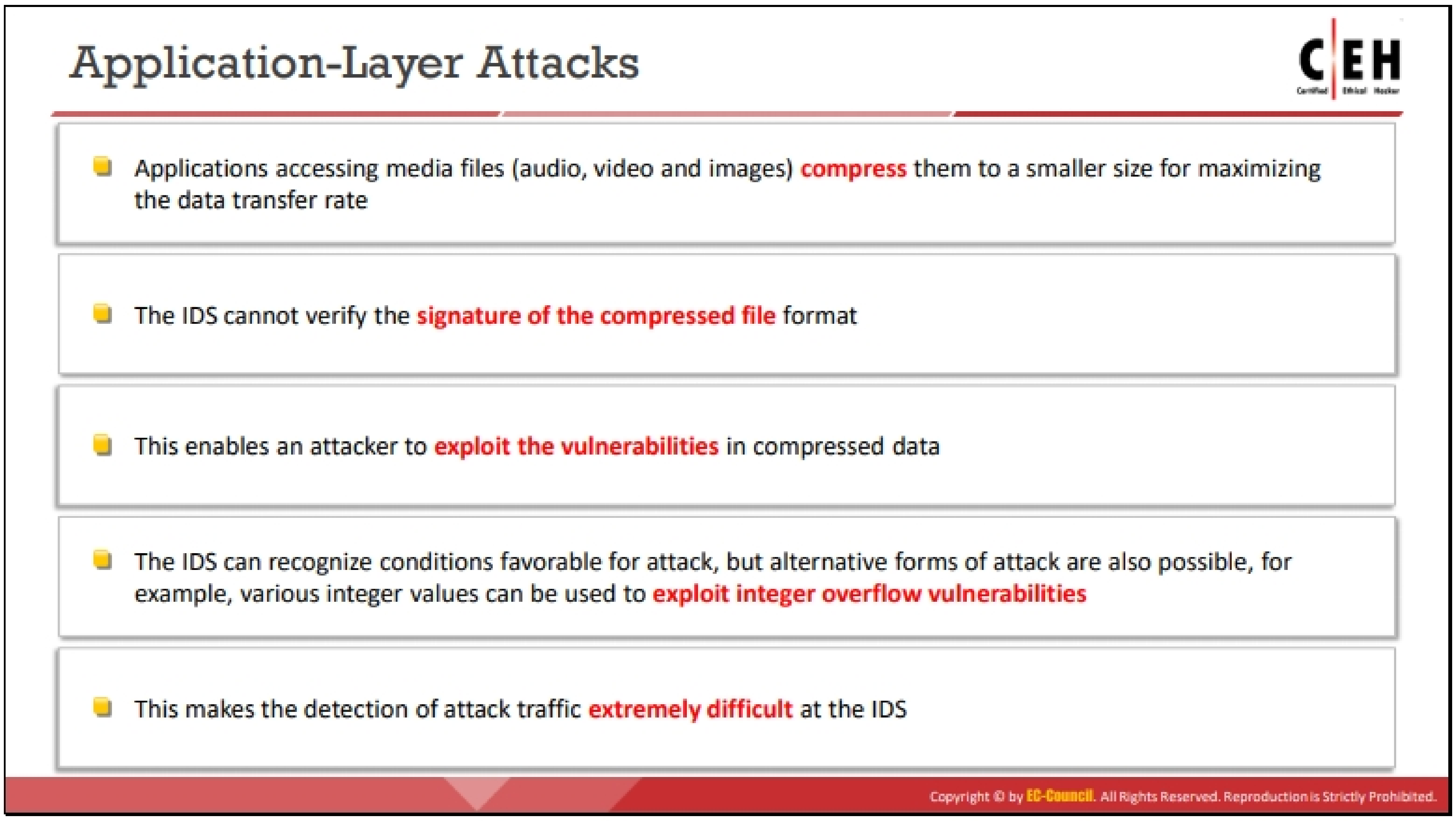
ASCII shellcodes contain only characters from the ASCII standard. Such shellcodes allow attackers to bypass commonly enforced character restrictions within the string input code. They also help attackers bypass IDS pattern matching signatures because they hide strings similarly to polymorphic shellcodes. The IDS pattern matching mechanism does not work efficiently with ASCII values.

Using ASCII for shellcode is very restrictive in that it limits what the shellcode can do under some circumstances, as not all assembly instructions convert directly into ASCII values. This restriction bypasses using other instructions, or a combination of instructions, which convert to ASCII character representation, serving the same purpose as those instructions that convert improperly.

An ASCII shellcode example is given below:

```
char shellcode[] =
"LLLLYhb0pLX5b0pLHSSPPWQPPaPWSUTBRDJfh5tDS"
"RajYX0Dka0TkafhN9fYf1Lkb0TkdjfY0Lkf0Tkgfh"
"6rfYf1Lki0tkkh95h8Y1LkmjpY0Lkq0tkrh2wnuX1"
"Dks0tkwjfX0Dkx0tkx0tkyCjnY0LkzC0TkzCCjtX0"
"DkzC0tkzCj3X0Dkz0TkzC0tkzChjG3IY1LkzCCCC0"
"tkzChpfcMX1DkzCCCC0tkzCh4pCnY1Lkz1TkzCCCC"
"fhJGfXf1Dkzf1tkzCCjHX0DkzCCCCjvY0LkzCCCjd"
"X0DkzC0TkzCjWX0Dkz0TkzCjdX0DkzCjXY0Lkz0tk"
"zMdgvvn9F1r8F55h8pG9wnuvjrNfrVx2LGkG3IDpf"
"cM2KgmnJGgbinYshdvD9d";
```

When executed, the shellcode above runs a "`/bin/sh`" shell. "`bin`" and "`sh`" are contained in the last few bytes of the shellcode.



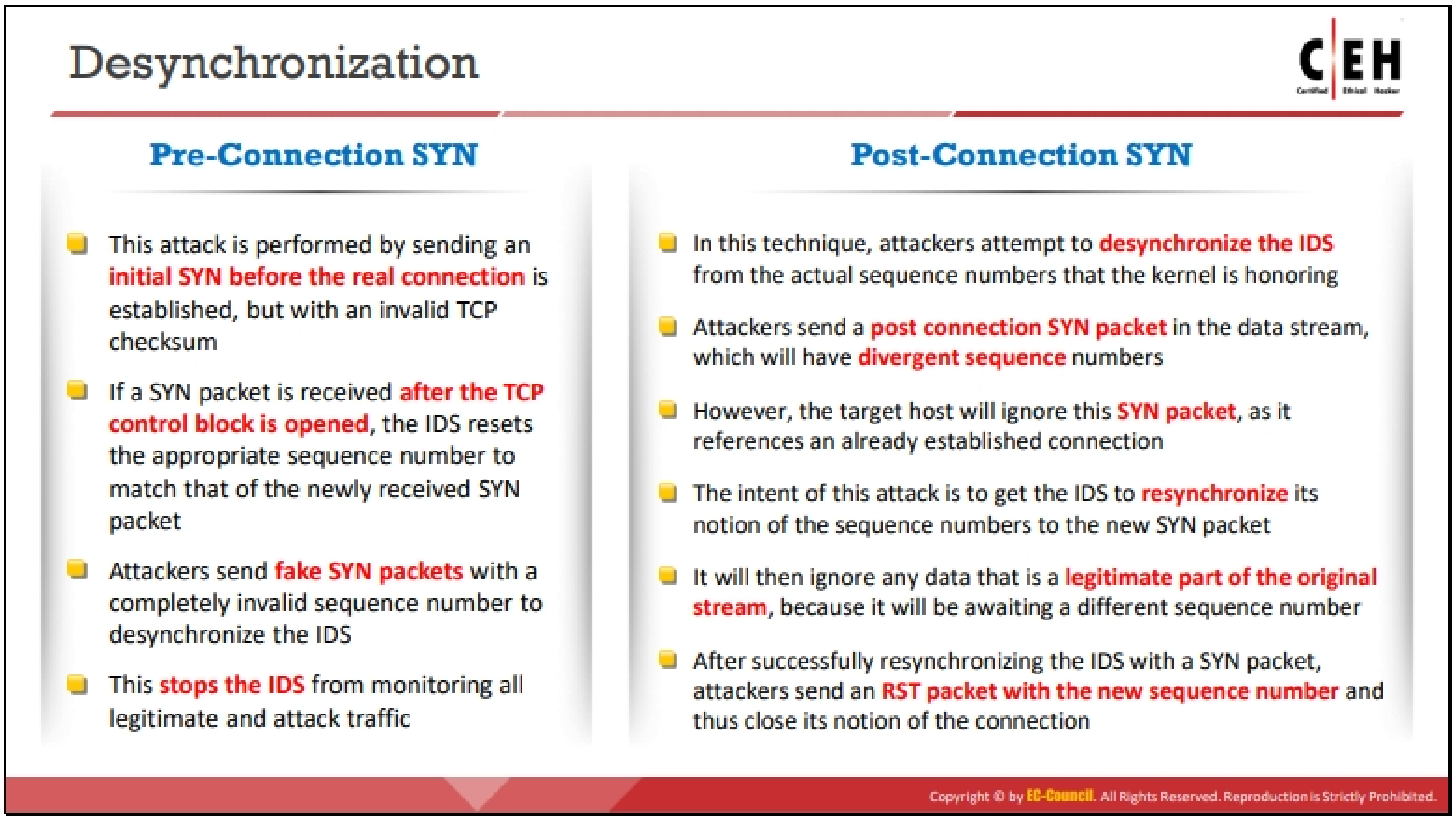
Application-Layer Attacks

- Applications accessing media files (audio, video and images) **compress** them to a smaller size for maximizing the data transfer rate
- The IDS cannot verify the **signature of the compressed file** format
- This enables an attacker to **exploit the vulnerabilities** in compressed data
- The IDS can recognize conditions favorable for attack, but alternative forms of attack are also possible, for example, various integer values can be used to **exploit integer overflow vulnerabilities**
- This makes the detection of attack traffic **extremely difficult** at the IDS

Application-Layer Attacks

Media files such as images, audios, and videos can be compressed so that they can be rapidly transferred as smaller chunks. Attackers find flaws in this compressed data and perform attacks; even the IDS signatures cannot identify the attack code within data thus compressed.

Many applications that deal with such media files employ some form of compression to increase the data transfer speed. When you find a flaw in these applications, the entire attack can occur within the compressed data, and the IDS will have no way to check the compressed file format for signatures. This enables an attacker to exploit the vulnerabilities in the compressed data. Many IDS look for specific conditions that allow for an attack. However, there are times when the attack can take many different forms. For example, attackers can exploit the integer overflow vulnerabilities using several different integer values. This fact, combined with compressed data, makes signature detection extremely difficult.



Desynchronization

Pre-Connection SYN

- This attack is performed by sending an **initial SYN before the real connection** is established, but with an invalid TCP checksum
- If a SYN packet is received **after the TCP control block is opened**, the IDS resets the appropriate sequence number to match that of the newly received SYN packet
- Attackers send **fake SYN packets** with a completely invalid sequence number to desynchronize the IDS
- This **stops the IDS** from monitoring all legitimate and attack traffic

Post-Connection SYN

- In this technique, attackers attempt to **desynchronize the IDS** from the actual sequence numbers that the kernel is honoring
- Attackers send a **post connection SYN packet** in the data stream, which will have **divergent sequence** numbers
- However, the target host will ignore this **SYN packet**, as it references an already established connection
- The intent of this attack is to get the IDS to **resynchronize** its notion of the sequence numbers to the new SYN packet
- It will then ignore any data that is a **legitimate part of the original stream**, because it will be awaiting a different sequence number
- After successfully resynchronizing the IDS with a SYN packet, attackers send an **RST packet with the new sequence number** and thus close its notion of the connection

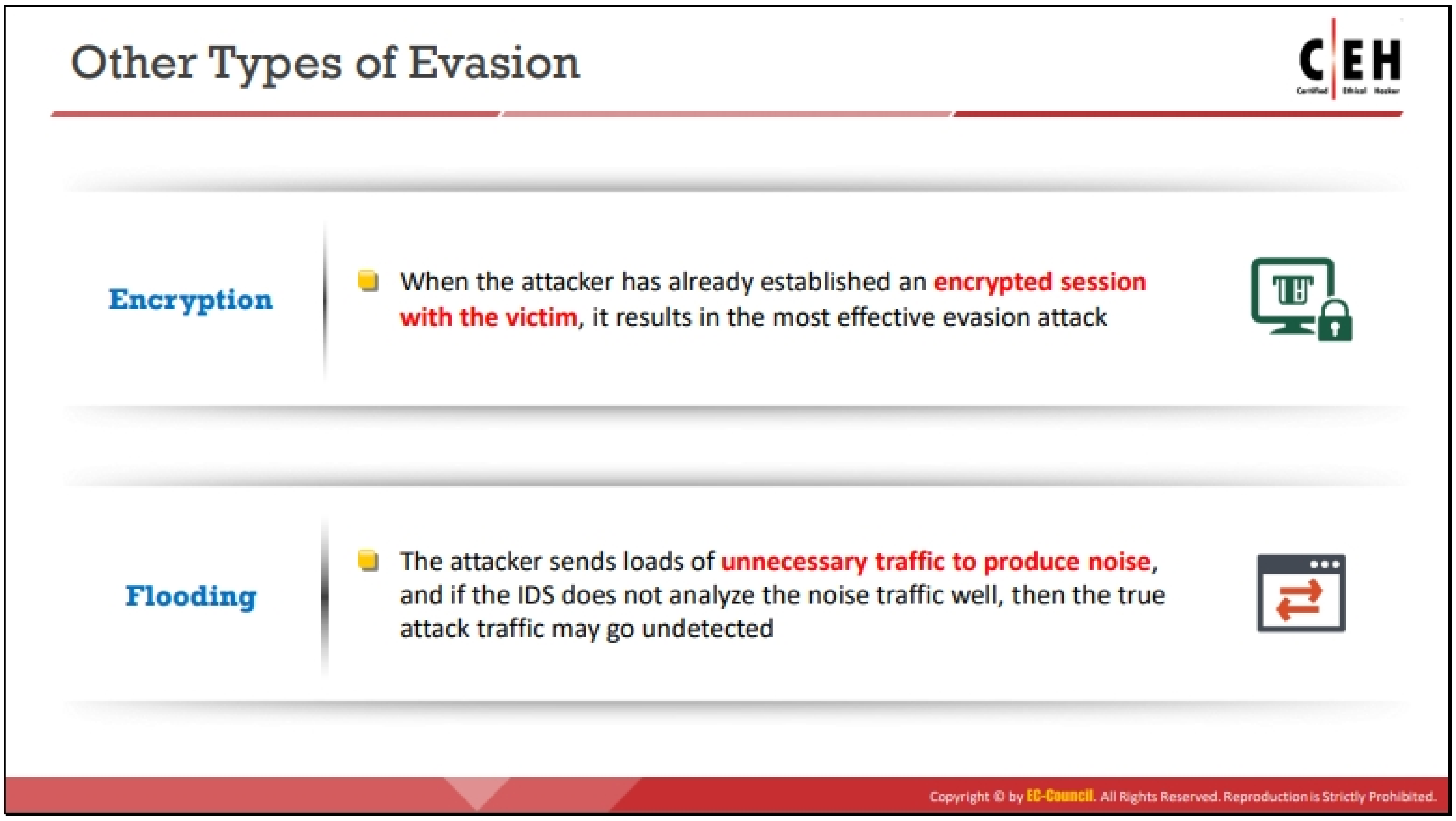
Desynchronization

- **Pre-Connection SYN**

  This attack is performed by sending an initial SYN before the real connection is established, but with an invalid TCP checksum. The IDS can ignore or accept subsequent SYNs in a connection. If a SYN packet is received after the TCP control block is opened, the IDS resets the appropriate sequence number to match the newly received SYN packet. Attackers send fake SYN packets with a completely invalid sequence number to desynchronize the IDS. This stops the IDS from monitoring all legitimate and attack traffic. If the IDS is smart, it does not check the TCP checksum. If the IDS checks the checksum, the attack is synchronized and a bogus sequence number is sent to the IDS before the real connection occurs.

- **Post-Connection SYN**

  In this technique, attackers attempt to desynchronize the IDS from the actual sequence numbers that the kernel is honoring. Send a post-connection SYN packet in the data stream, which will have divergent sequence numbers but otherwise meet all the necessary criteria to be accepted by the target host. However, the target host will ignore this SYN packet, as it references an already established connection. This attack intends to get the IDS to resynchronize its notion of the sequence numbers to the new SYN packet. It will then ignore any data that is a legitimate part of the original stream because it will be waiting for a different sequence number. Once you succeed in resynchronizing the IDS with a SYN packet, send an RST packet with the new sequence number and close down its notion of the connection.



Other Types of Evasion

Encryption	When the attacker has already established an **encrypted session with the victim**, it results in the most effective evasion attack
Flooding	The attacker sends loads of **unnecessary traffic to produce noise**, and if the IDS does not analyze the noise traffic well, then the true attack traffic may go undetected

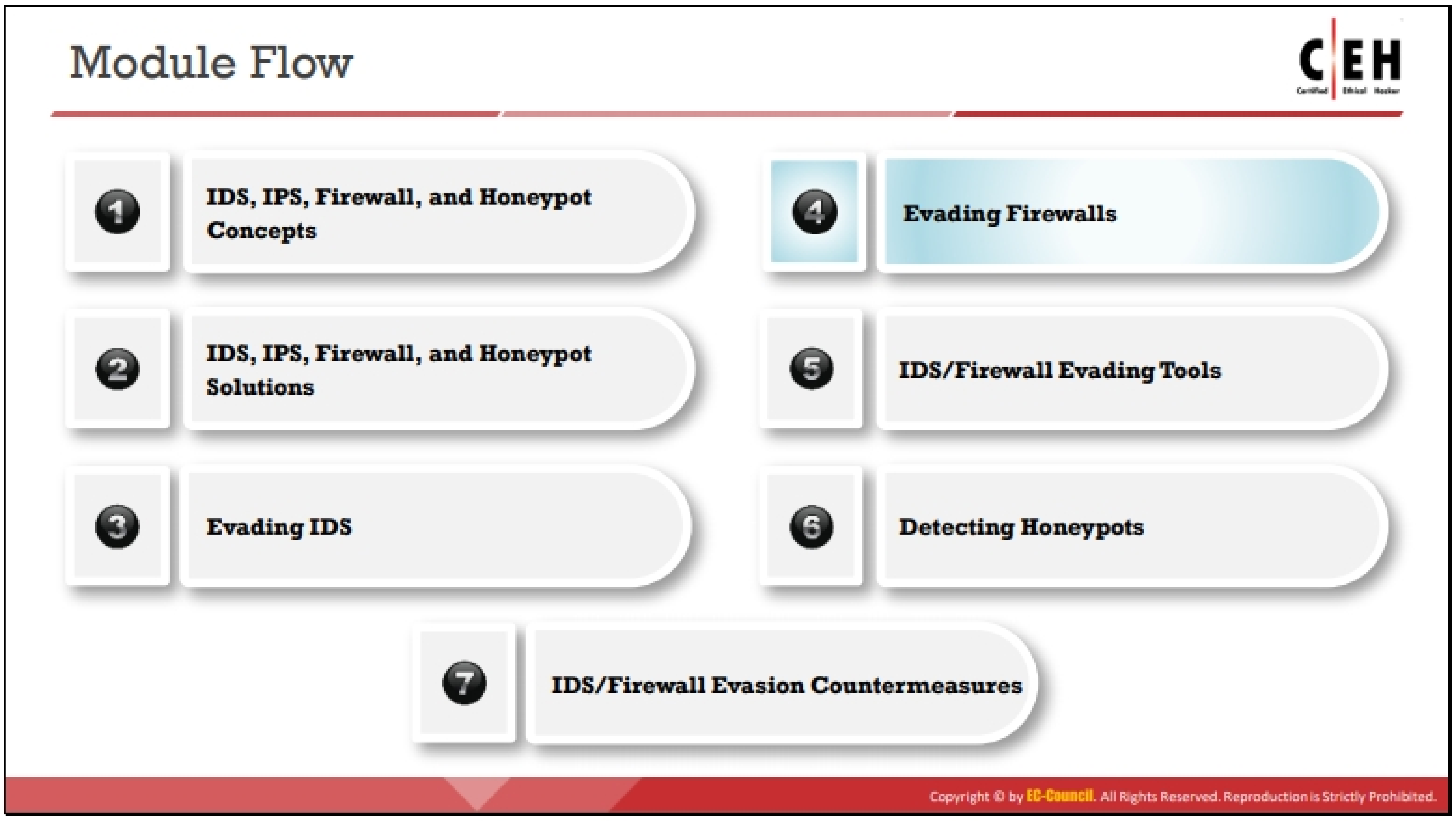
Other Types of Evasion

- **Encryption**

  Network-based intrusion detection analyzes traffic in the network from the source to the destination. If an attacker succeeds in establishing an encrypted session with his/her target host using a secure shell (SSH), secure socket layer (SSL), or virtual private network (VPN) tunnel, the IDS will not analyze the packets going through these encrypted communications. Thus, an attacker sends malicious traffic using such secure channels, thereby evading IDS security.

- **Flooding**

  IDS use resources such as memory and processor speed to analyze the traffic going through them. To bypass IDS security, attackers flood IDS resources with noise or fake traffic to exhaust them with having to analyze flooded traffic. Once such attacks succeed, attackers send malicious traffic toward the target system behind the IDS, which offers little or no intervention. Thus, true attack traffic might go undetected.

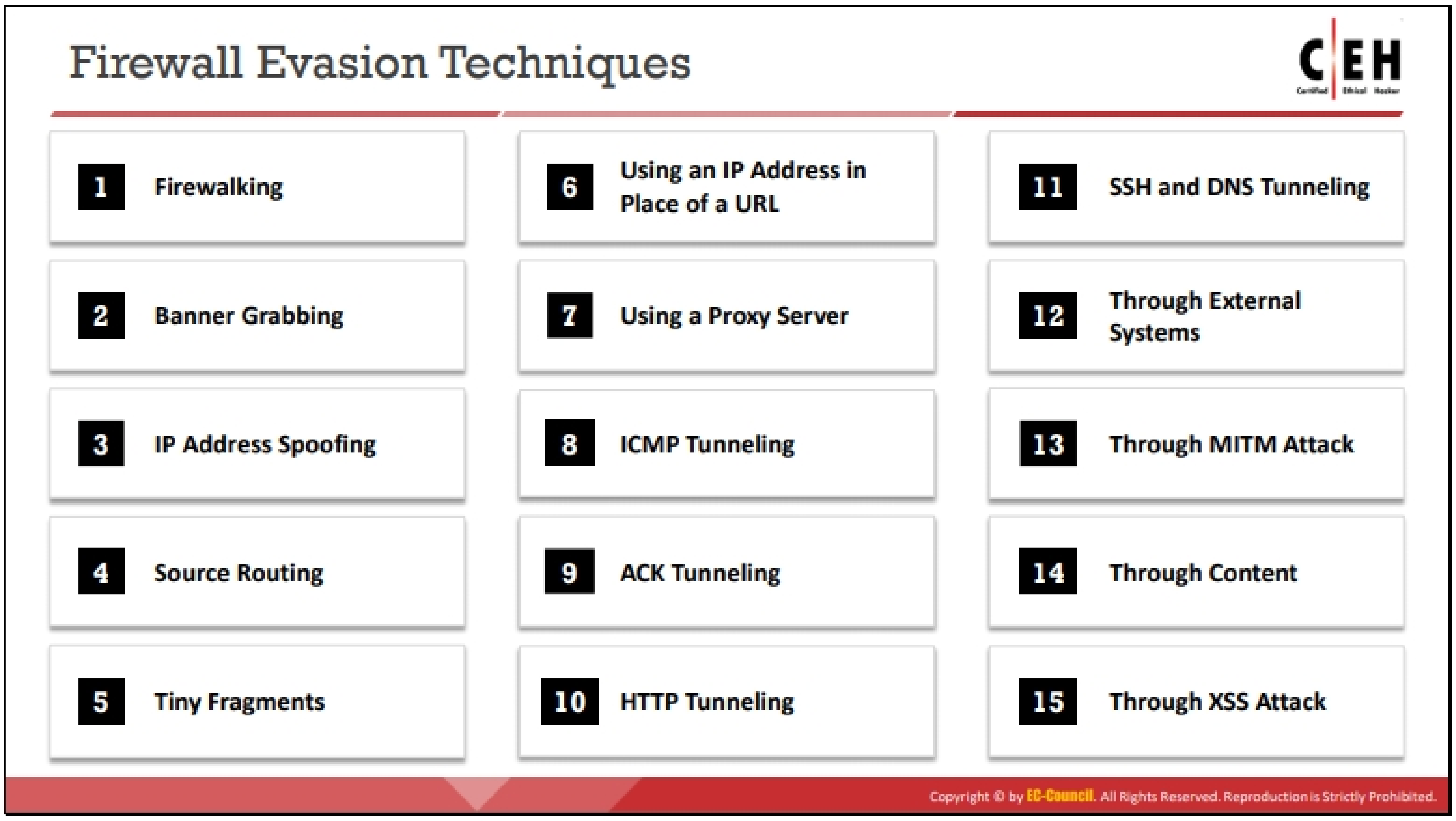


Module Flow

1. IDS, IPS, Firewall, and Honeypot Concepts
2. IDS, IPS, Firewall, and Honeypot Solutions
3. Evading IDS
4. Evading Firewalls
5. IDS/Firewall Evading Tools
6. Detecting Honeypots
7. IDS/Firewall Evasion Countermeasures

Evading Firewalls

The previous section explained how attackers use various techniques to bypass IDS. Similarly, they can also use various tricks and techniques to bypass firewalls. This section discusses the different techniques used by attackers to bypass firewall security.



Firewall Evasion Techniques

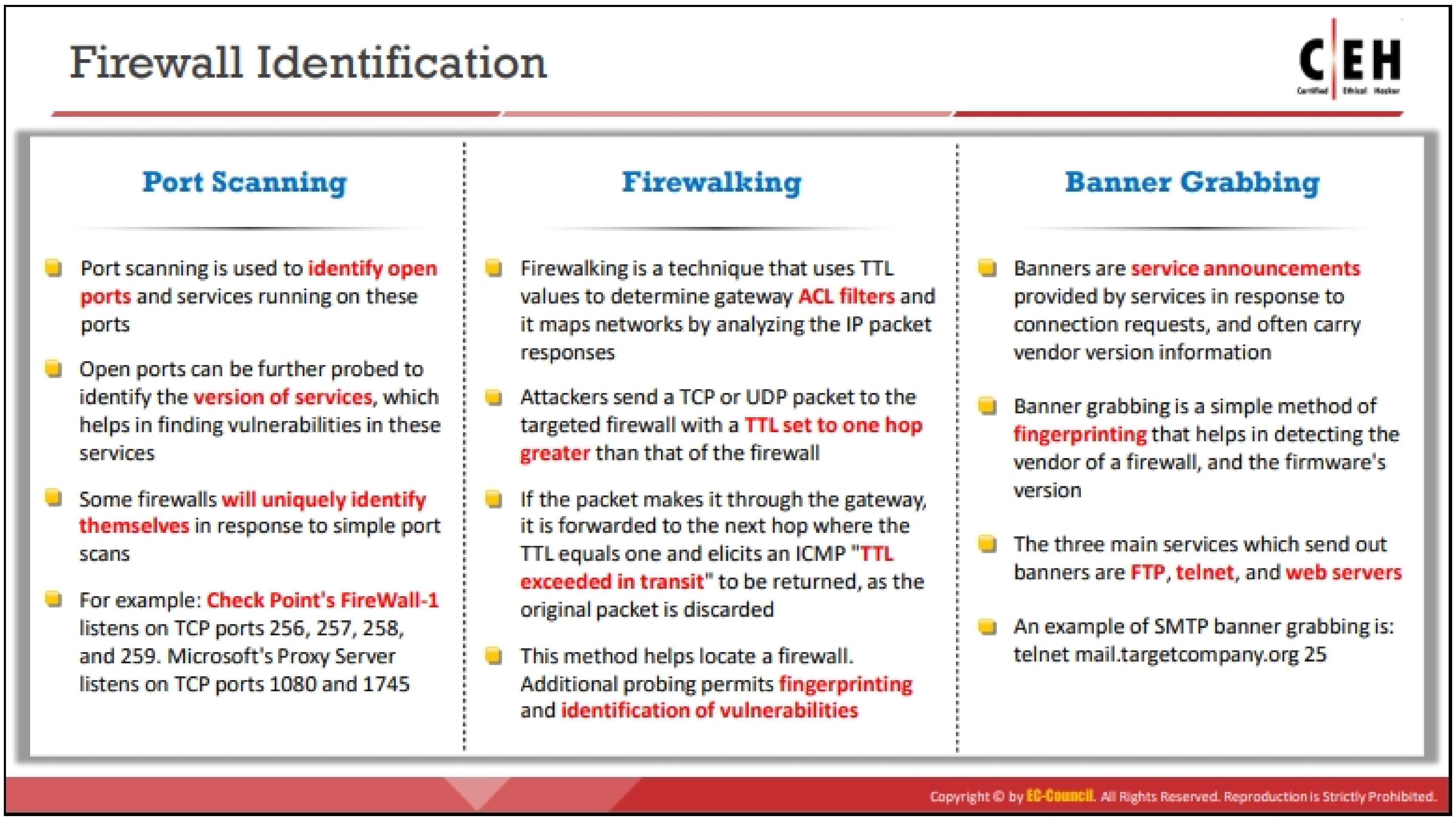
1. Firewalking
2. Banner Grabbing
3. IP Address Spoofing
4. Source Routing
5. Tiny Fragments
6. Using an IP Address in Place of a URL
7. Using a Proxy Server
8. ICMP Tunneling
9. ACK Tunneling
10. HTTP Tunneling
11. SSH and DNS Tunneling
12. Through External Systems
13. Through MITM Attack
14. Through Content
15. Through XSS Attack

Firewall Evasion Techniques

Bypassing a firewall is a technique whereby an attacker manipulates the attack sequence to avoid being detected by the underlying security firewall. The firewall operates on a predefined set of rules, and with thorough knowledge and skill, an attacker can bypass the firewall by employing various firewall bypassing techniques. Using these techniques, the attacker tricks the firewall into not filtering the malicious traffic generated by the attacker.

Some firewall bypassing techniques are as follows:

- Port Scanning
- Firewalking
- Banner Grabbing
- IP Address Spoofing
- Source Routing
- Tiny Fragments
- Using an IP Address in Place of a URL
- Using Anonymous Website Surfing Sites
- Using a Proxy Server
- ICMP Tunneling
- ACK Tunneling
- HTTP Tunneling
- SSH Tunneling
- DNS Tunneling
- Through External Systems
- Through MITM Attack
- Through Content
- Through XSS Attack



Firewall Identification

Port Scanning

- Port scanning is used to **identify open ports** and services running on these ports
- Open ports can be further probed to identify the **version of services**, which helps in finding vulnerabilities in these services
- Some firewalls **will uniquely identify themselves** in response to simple port scans
- For example: **Check Point's FireWall-1** listens on TCP ports 256, 257, 258, and 259. Microsoft's Proxy Server listens on TCP ports 1080 and 1745

Firewalking

- Firewalking is a technique that uses TTL values to determine gateway **ACL filters** and it maps networks by analyzing the IP packet responses
- Attackers send a TCP or UDP packet to the targeted firewall with a **TTL set to one hop greater** than that of the firewall
- If the packet makes it through the gateway, it is forwarded to the next hop where the TTL equals one and elicits an ICMP "**TTL exceeded in transit**" to be returned, as the original packet is discarded
- This method helps locate a firewall. Additional probing permits **fingerprinting** and **identification of vulnerabilities**

Banner Grabbing

- Banners are **service announcements** provided by services in response to connection requests, and often carry vendor version information
- Banner grabbing is a simple method of **fingerprinting** that helps in detecting the vendor of a firewall, and the firmware's version
- The three main services which send out banners are **FTP**, **telnet**, and **web servers**
- An example of SMTP banner grabbing is: telnet mail.targetcompany.org 25

Firewall Identification

- **Port Scanning**

  Ports are points from which computers send or accept information from network resources. Port scanning is used to identify open ports and the services running on these ports. Finding open ports is an attacker's first step toward gaining access to the target system. To do so, the attacker systematically scans the target's ports to identify the versions of services, which helps in finding vulnerabilities in these services. Attackers sometimes use automated port-scanning utilities to do so, many of which are easily available.

  How Attackers Scan Ports

  Port scanning consists of sending messages to each port, one at a time. The kind of response received indicates whether the system is using the port, leaving it exposed to the discovery of weaknesses. Some firewalls will uniquely identify themselves using simple port scans. For example, Check Point's FireWall-1 listens on TCP ports 256, 257, 258, and 259, and Microsoft's Proxy Server usually listens on TCP ports 1080 and 1745.

- **Firewalking**

  Firewalking is a method of collecting information about remote networks behind firewalls. It is a technique that uses TTL values to determine gateway ACL filters and map networks by analyzing the IP packet response. It probes ACLs on packet filtering routers/firewalls using the same method as tracerouting. Firewalking involves sending TCP or UDP packets into the firewall where the TTL value is one hop greater than the targeted firewall. If the packet makes it through the gateway, the system forwards it to the next hop, where the TTL equals one, and prompts an ICMP error message at the point of rejection with a "TTL exceeded in transit" message. This method helps locate a firewall; additional probing facilitates fingerprinting and identification of vulnerabilities.

Firewalk is a well-known application used for firewalking. It has two phases: a network discovery phase and a scanning phase. It comes with various open-source Linux distributions. Nmap has a firewalk script that can be used to perform firewalking.

- **Banner Grabbing**

  Banners are service announcements provided by services in response to connection requests, and they often carry vendor version information. Banner grabbing is a simple method of fingerprinting that helps in detecting the vendor of a firewall and the firmware version. It identifies the service running on the system. Attackers use banner grabbing to fingerprint services and thus discover the services running on firewalls. The three primary services that send out banners are FTP, Telnet, and web servers.

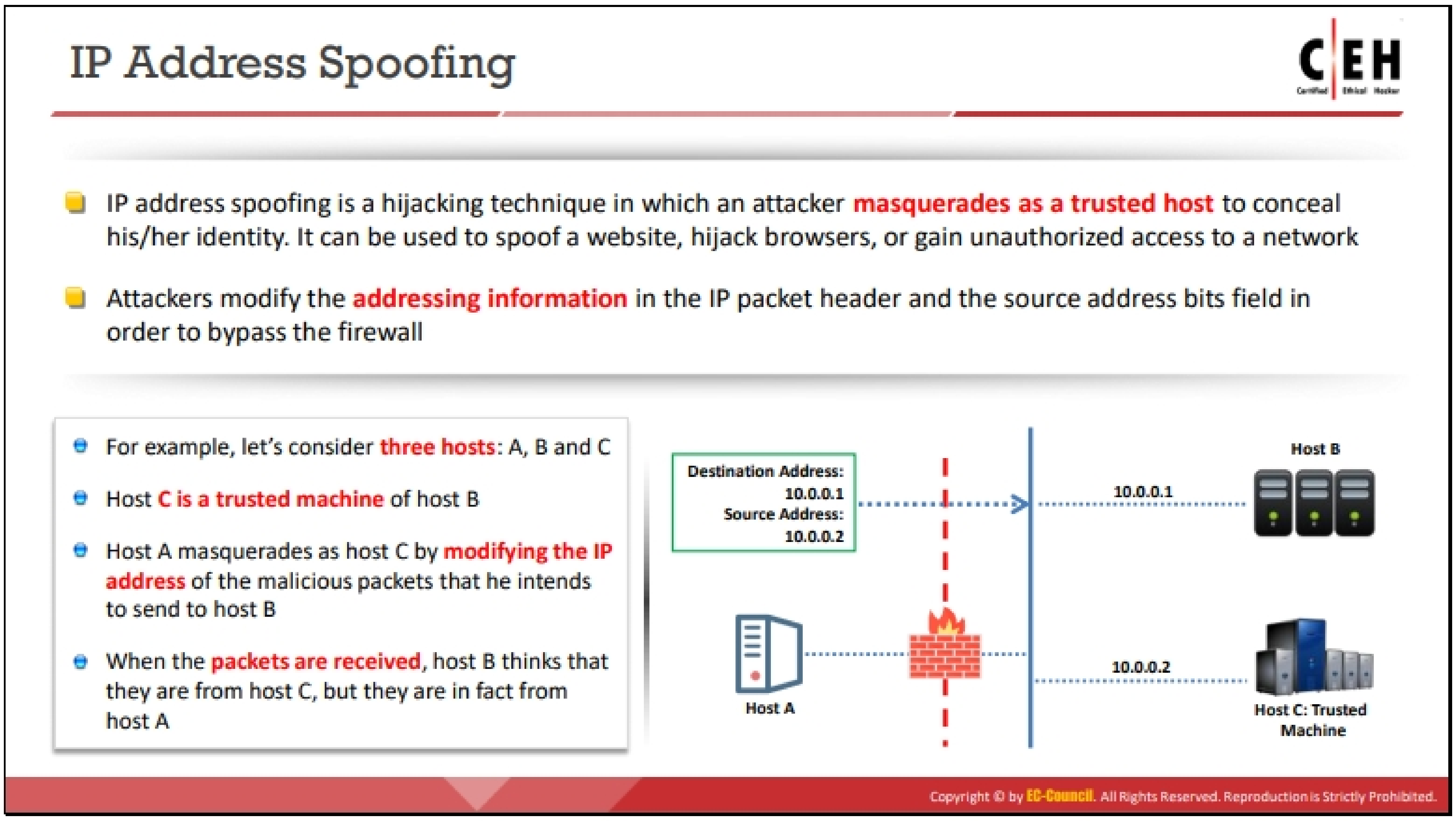
  A firewall does not block banner grabbing because the connection between the attacker's system and the target system appears legitimate. An example of SMTP banner grabbing is `telnet mail.targetcompany.org 25`.

  The syntax is "`<service name > <service running > <port number>`"

  Banner grabbing is used for specifying banners and application information. For example, when the user opens a telnet connection to a known port on the target server and presses Enter a few times, if required, it displays the following result:

```
C:\>telnet www.corleone.com 80

HTTP/1.0 400 Bad Request

Server: Netscape – Commerce/1.12
```

  This system works with many other common applications that respond to a set port. The information generated through banner grabbing can boost the attacker's efforts to further compromise the system. With information about the version and the vendor of the web server, the attacker can further focus on employing platform-specific exploit techniques. Services on ports such as FTP, Telnet, and web servers should not remain open, as they are vulnerable to banner grabbing.



IP Address Spoofing

- IP address spoofing is a hijacking technique in which an attacker **masquerades as a trusted host** to conceal his/her identity. It can be used to spoof a website, hijack browsers, or gain unauthorized access to a network
- Attackers modify the **addressing information** in the IP packet header and the source address bits field in order to bypass the firewall

- For example, let's consider **three hosts**: A, B and C
- Host **C is a trusted machine** of host B
- Host A masquerades as host C by **modifying the IP address** of the malicious packets that he intends to send to host B
- When the **packets are received**, host B thinks that they are from host C, but they are in fact from host A

IP Address Spoofing

Most firewalls filter packets based on the source IP address. These firewalls examine the source IP address and determine whether the packet is coming from a legitimate source or an illegitimate source. The IDS filters packets from illegitimate sources. Attackers use the IP spoofing technique to bypass such firewalls.

IP address spoofing is a hijacking technique in which an attacker masquerades as a trusted host to conceal his identity, spoof a website, hijack browsers, or gain unauthorized access to a network. In IP spoofing, the attacker creates IP packets by using a forged IP address and gains access to the system or network without authorization. Attackers modify the addressing information in the IP packet header and the source address bits field to bypass the firewall. The attacker spoofs the message; therefore, the destination host thinks that it has come from a reliable source. Thus, the attacker succeeds in impersonating others with the help of IP spoofing. Hackers use this technique to avoid detection during spamming and various other activities.

For example, let us consider three hosts: A, B, and C. Host C is a trusted machine of host B. Host A masquerades as host C by modifying the IP address of the malicious packets that it intends to send to host B. When the packets are received, host B thinks that they are from host C, but they are actually from host A.

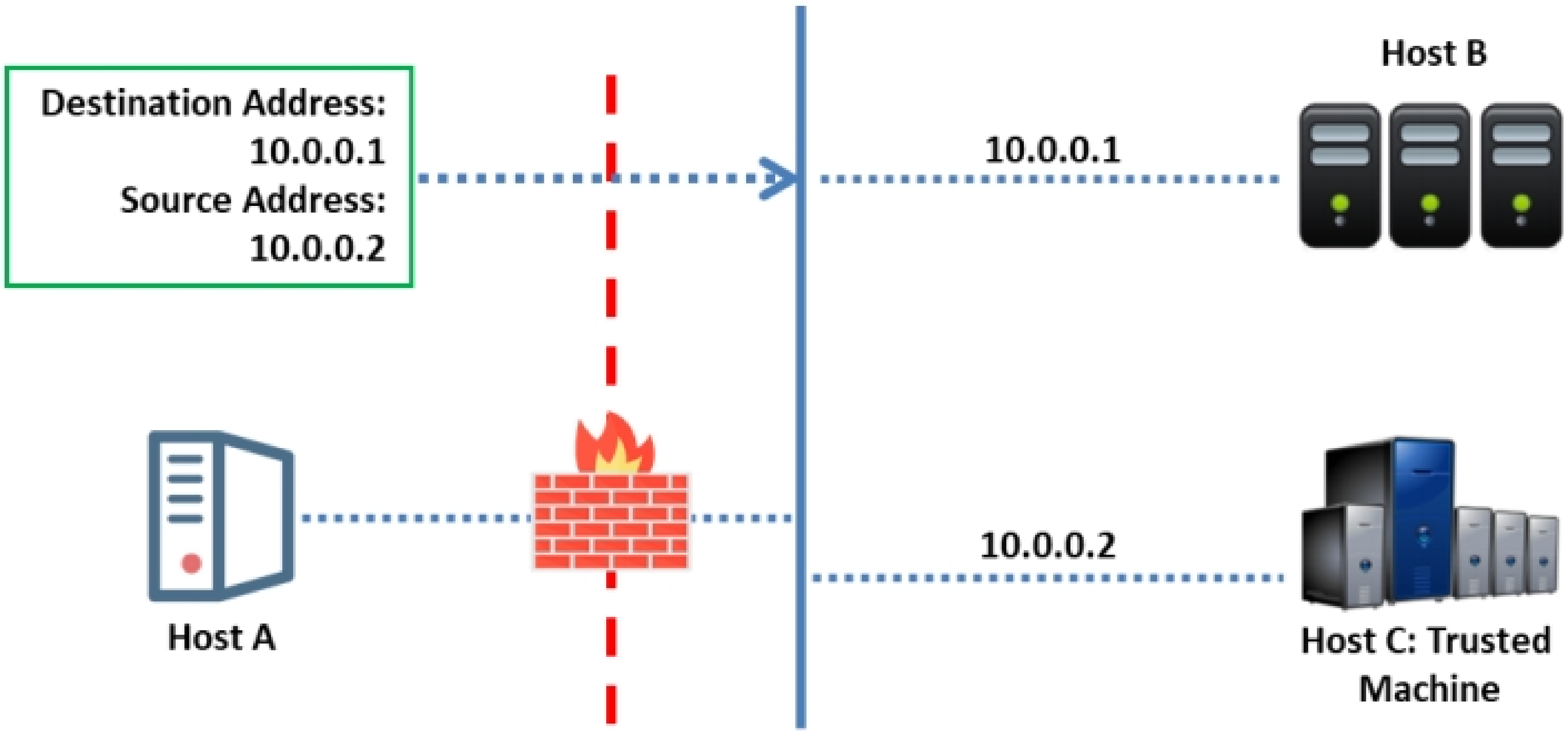
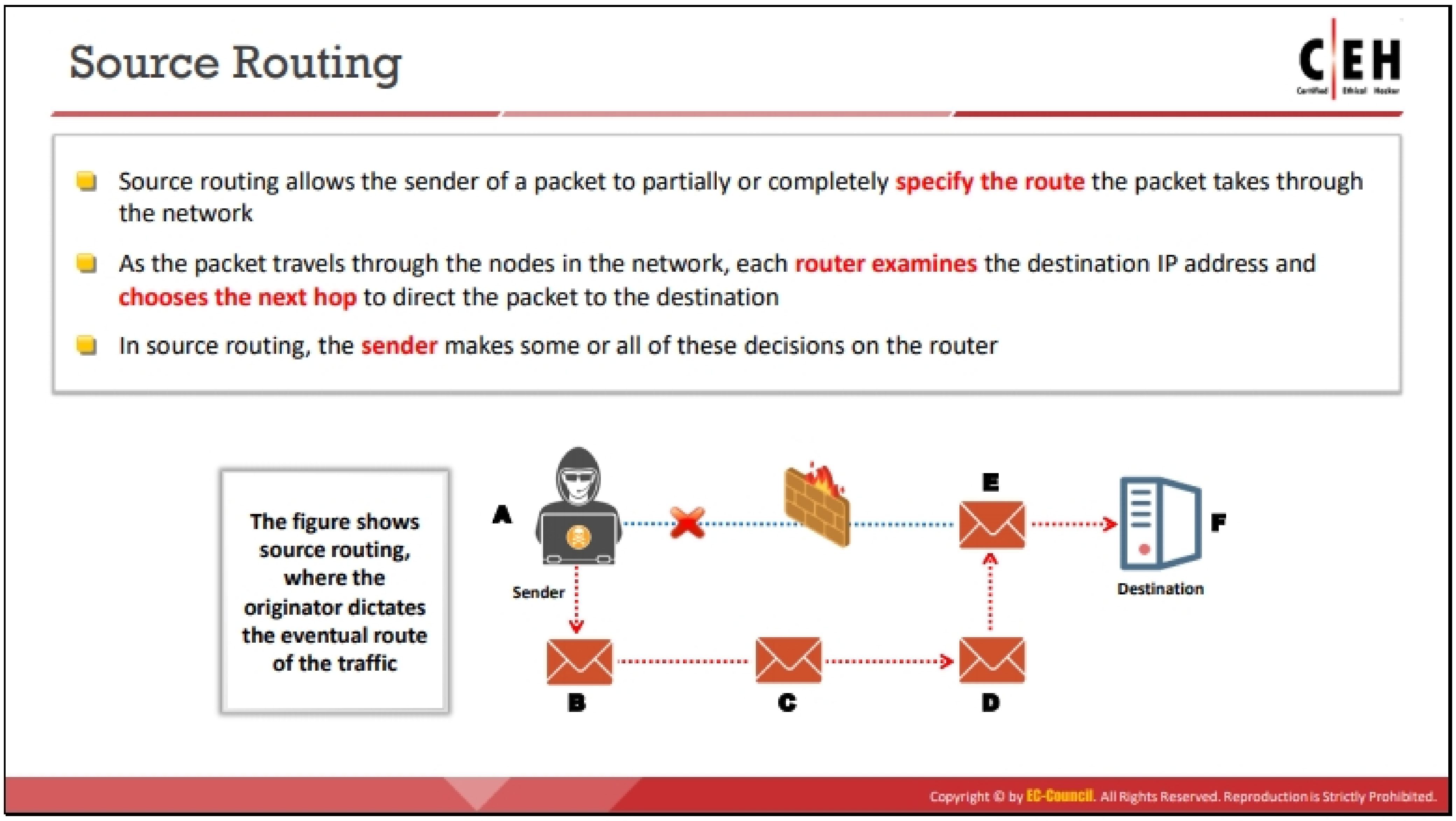

Destination Address:
10.0.0.1
Source Address:
10.0.0.2
Host B
10.0.0.1
Host A
10.0.0.2
Host C: Trusted Machine

Figure 12.40: Evading Firewall using IP Address Spoofing



Source Routing

- Source routing allows the sender of a packet to partially or completely **specify the route** the packet takes through the network
- As the packet travels through the nodes in the network, each **router examines** the destination IP address and **chooses the next hop** to direct the packet to the destination
- In source routing, the **sender** makes some or all of these decisions on the router

The figure shows source routing, where the originator dictates the eventual route of the traffic

Source Routing

Using this technique, the sender of the packet designates the route (partially or entirely) that a packet should take through the network such that the designated route should bypass the firewall node. Thus, the attacker can evade firewall restrictions.

When these packets travel through the network nodes, each router examines the destination IP address and chooses the next hop to direct the packet to the destination. In source routing, the sender makes some or all of these decisions on the router.

Source routing is categorized into two approaches: loose source routing and strict source routing. In loose source routing, the sender specifies one or more stages that the packet must go through, whereas in strict source routing, the sender specifies the exact route the packet must go through.

The figure below shows source routing, where the originator dictates the eventual route of the traffic.

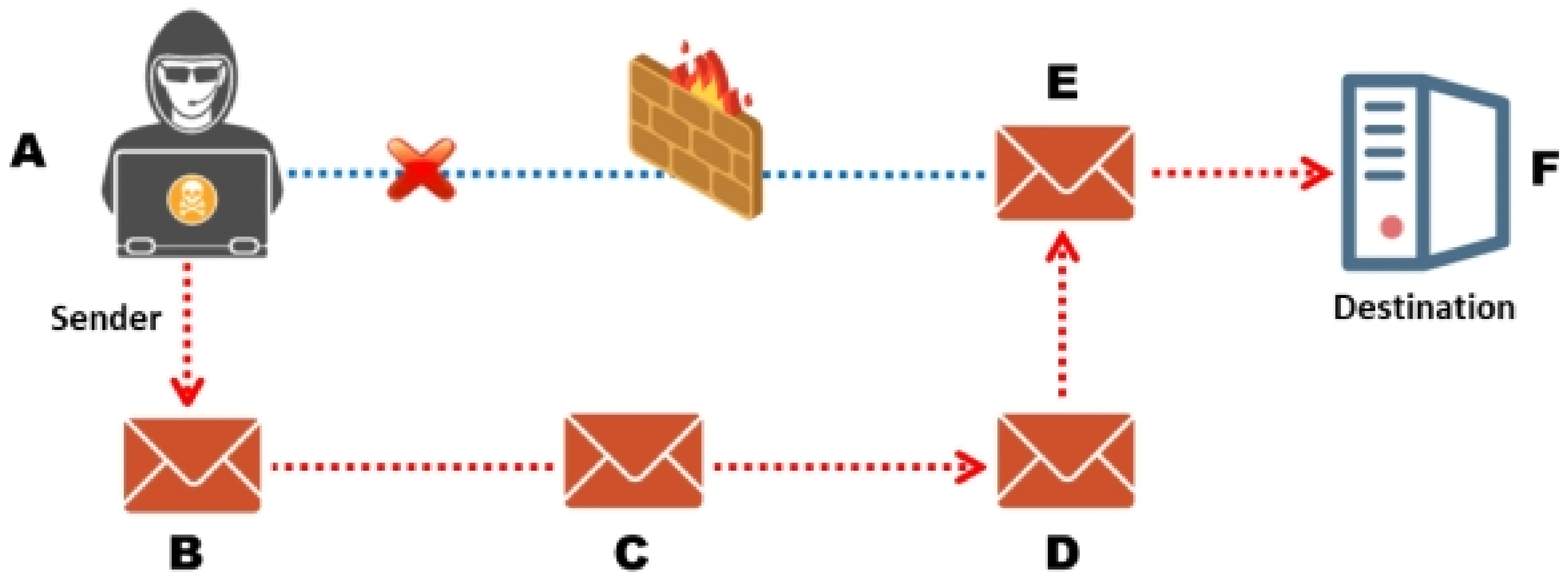
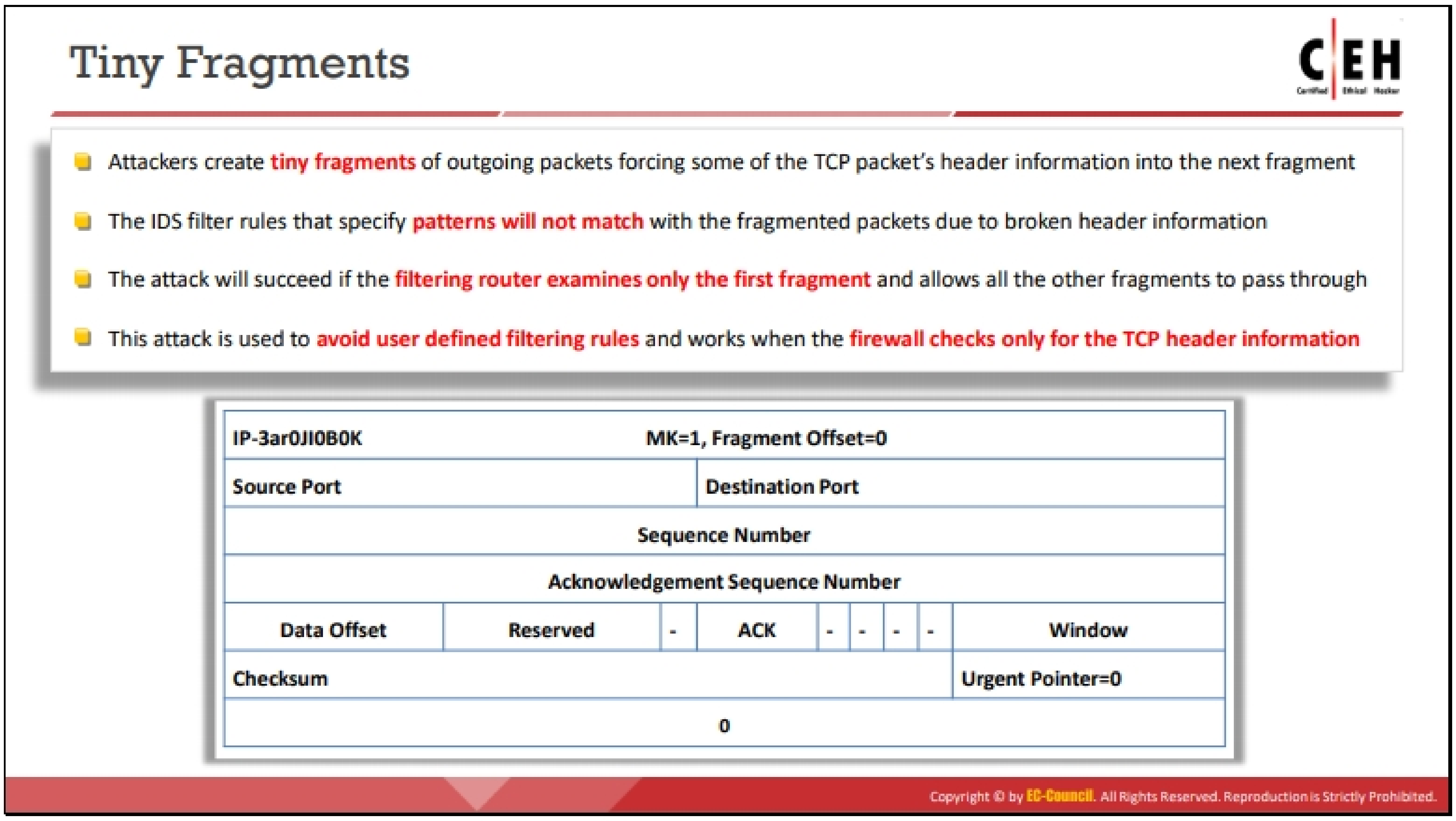


Figure 12.41: Evading Firewall using Source Routing



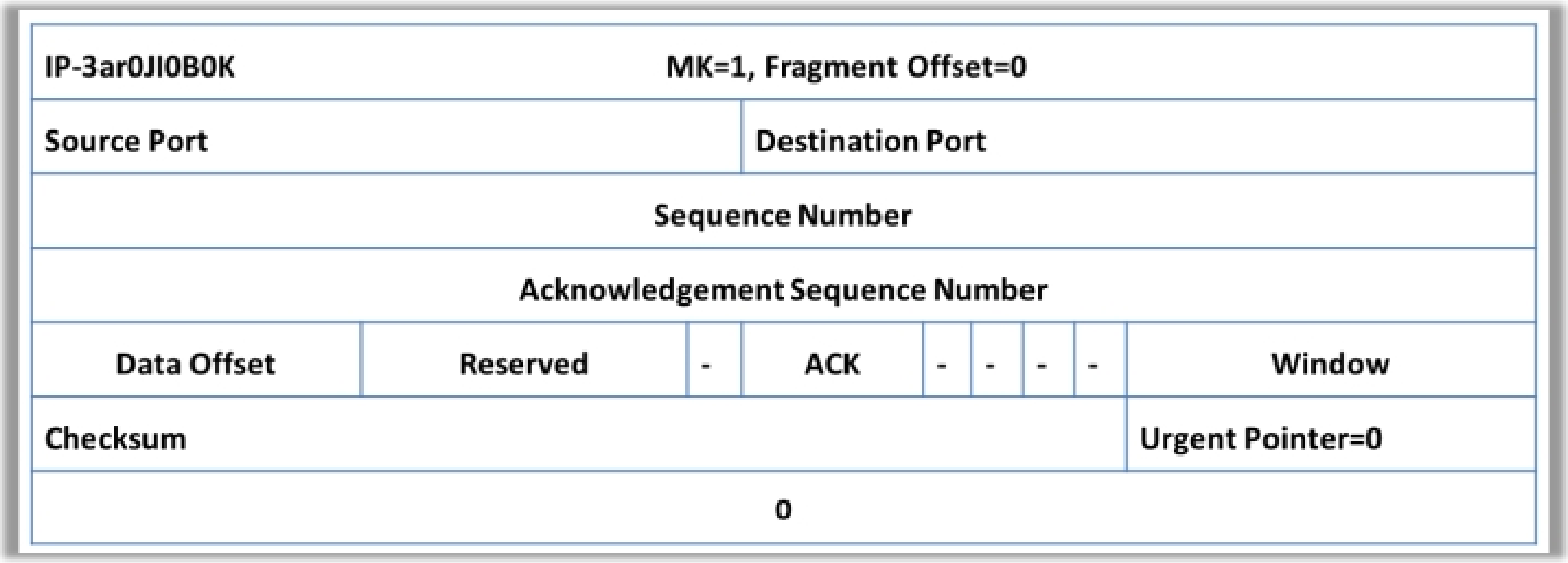
Tiny Fragments

- Attackers create **tiny fragments** of outgoing packets forcing some of the TCP packet's header information into the next fragment
- The IDS filter rules that specify **patterns will not match** with the fragmented packets due to broken header information
- The attack will succeed if the **filtering router examines only the first fragment** and allows all the other fragments to pass through
- This attack is used to **avoid user defined filtering rules** and works when the **firewall checks only for the TCP header information**

IP-3ar0JI0B0K			MK=1, Fragment Offset=0					
Source Port			Destination Port					
Sequence Number								
Acknowledgement Sequence Number								
Data Offset	Reserved	-	ACK	-	-	-	-	Window
Checksum								Urgent Pointer=0
0								

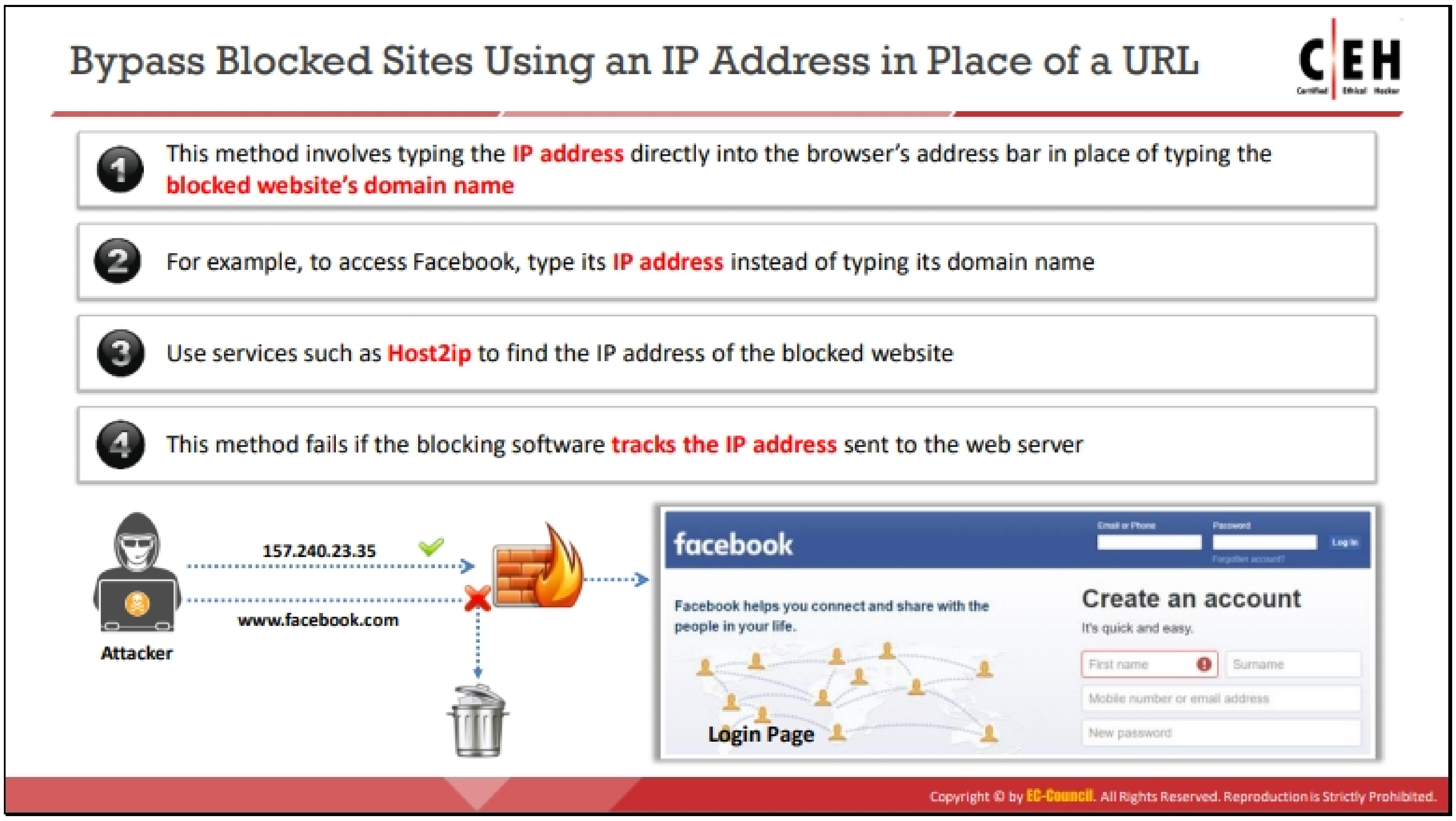
Tiny Fragments

Attackers create tiny fragments of outgoing packets, forcing some of the TCP packet's header information into the next fragment. The IDS filter rules that specify patterns will not match with the fragmented packets owing to the broken header information. The attack will succeed if the filtering router examines only the first fragment and allows all the other fragments to pass through. This attack is used to avoid user-defined filtering rules and works when the firewall checks only for the TCP header information.



IP-3ar0JI0B0K			MK=1, Fragment Offset=0					
Source Port			Destination Port					
Sequence Number								
Acknowledgement Sequence Number								
Data Offset	Reserved	-	ACK	-	-	-	-	Window
Checksum								Urgent Pointer=0
0								

Figure 12.42: TCP header format



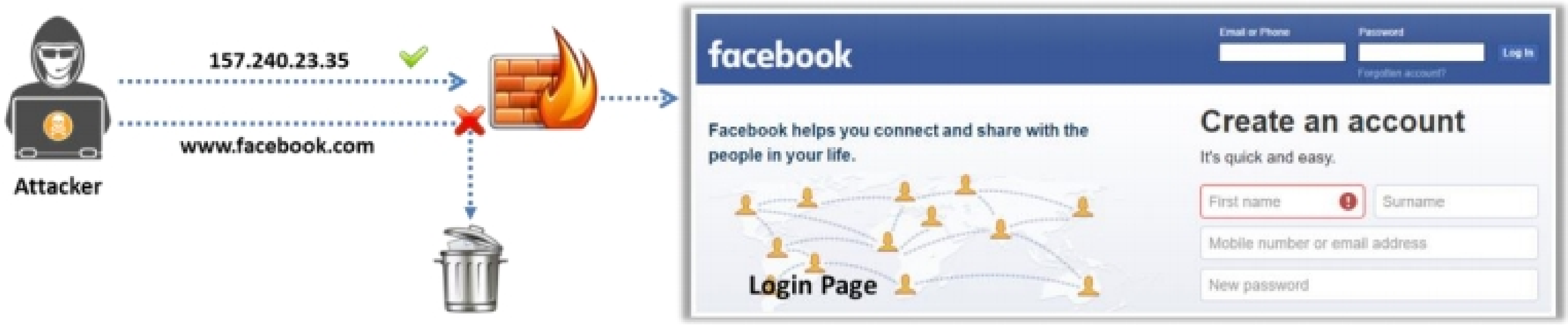
Bypass Blocked Sites Using an IP Address in Place of a URL

1. This method involves typing the **IP address** directly into the browser's address bar in place of typing the **blocked website's domain name**
2. For example, to access Facebook, type its **IP address** instead of typing its domain name
3. Use services such as **Host2ip** to find the IP address of the blocked website
4. This method fails if the blocking software **tracks the IP address** sent to the web server

157.240.23.35
www.facebook.com
Attacker
facebook
Login Page

Bypass Blocked Sites Using an IP Address in Place of a URL

This method involves typing a blocked website's IP address directly in the browser's address bar instead of the domain name. For example, to access Facebook, type its IP address instead of typing its domain name. Use services such as Host2ip to find the IP address of the blocked website. This method fails if the blocking software tracks the IP address sent to the web server.



157.240.23.35
www.facebook.com
Attacker
facebook
Email or Phone
Password
Log In
Forgotten account?
Login Page

Facebook helps you connect and share with the people in your life.

Create an account

It's quick and easy.

First name
Surname
Mobile number or email address
New password

Figure 12.43: Bypass blocked sites using the IP address instead of the URL

Bypass Blocked Sites Using Anonymous Website Surfing Sites

- There are many online anonymizer services that enable anonymous **surfing on the Internet**
- Some websites provide options to **encrypt the URL's** of the websites
- These services **hide the actual IP address of the surfer** and enable bypassing the IP-based firewall filter rules

Anonymizers

1. *https://www.anonymizer.com*
2. *https://www.free-proxy.com*
3. *https://anonymous-proxy-servers.net*
4. *https://zendproxy.com*
5. *https://proxify.com*
6. *http://www.guardster.com*
7. *http://anonymouse.org*
8. *https://www.boomproxy.com*
9. *http://ww7.anype.com*
10. *https://www.spysurfing.com*

Bypass Blocked Sites Using Anonymous Website Surfing Sites

Anonymous web-surfing sites help to browse the Internet anonymously and unblock blocked sites (i.e., evade firewall restrictions). By using these sites, you can surf restricted sites anonymously without revealing your IP address. Various anonymous web-surfing sites available, some of which provide options to encrypt website URLs.

The following is the list of proxy servers that can help you to access blocked websites. These proxy websites will hide the actual IP address and show another IP address, which could prevent the website from being blocked, thereby allowing access.

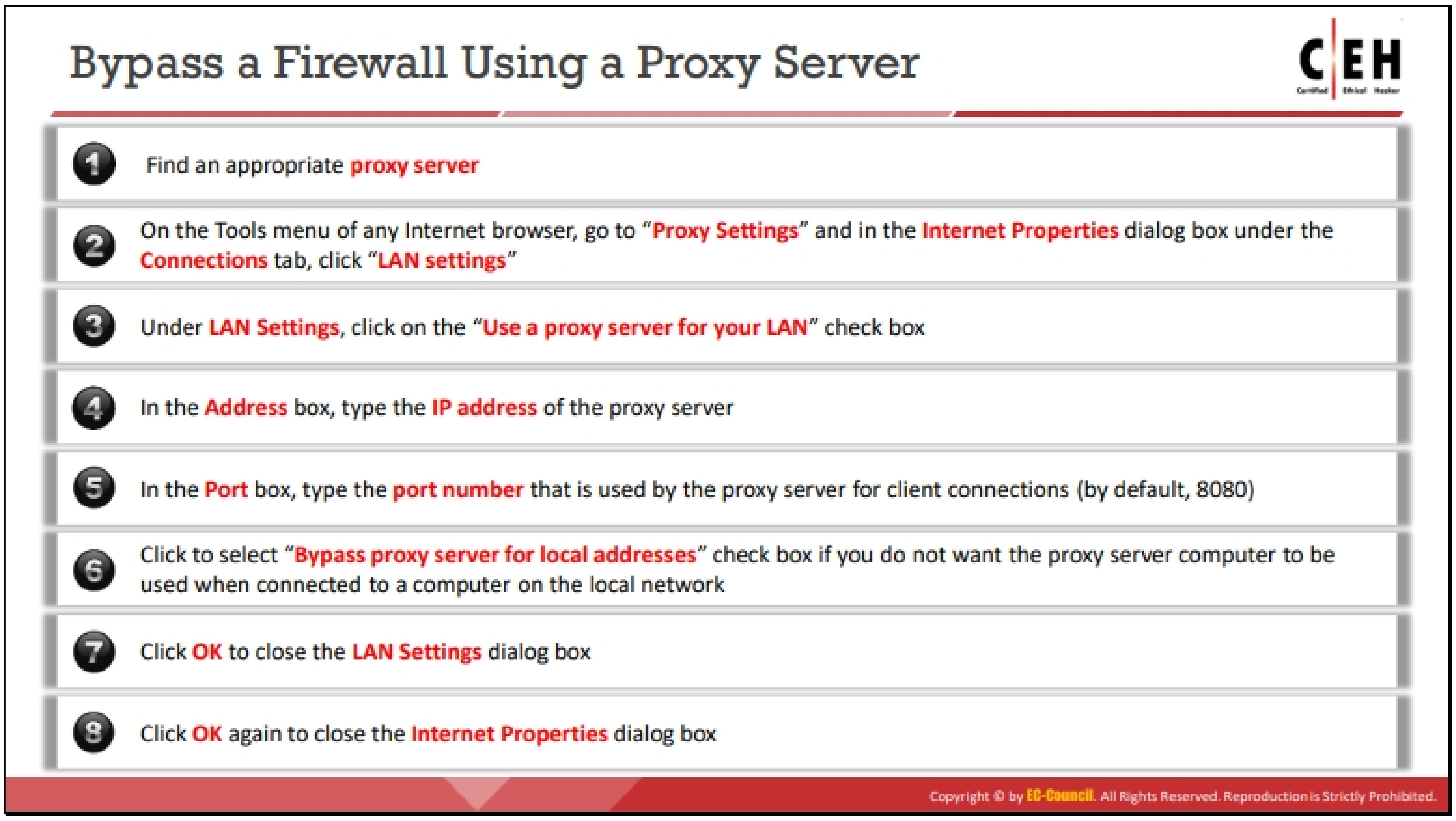
Anonymizer

Source: *https://www.anonymizer.com*

Anonymizer's VPN routes all the traffic through an encrypted tunnel directly from your laptop to secure and harden servers and networks. It then masks the real IP address to ensure complete and continuous anonymity for all online activities.

Some online anonymizers include:

- *https://www.free-proxy.com*
- *https://anonymous-proxy-servers.net*
- *https://zendproxy.com*
- *https://proxify.com*
- *http://www.guardster.com*
- *http://anonymouse.org*
- *https://www.boomproxy.com*
- *http://ww7.anype.com*
- *https://www.spysurfing.com*



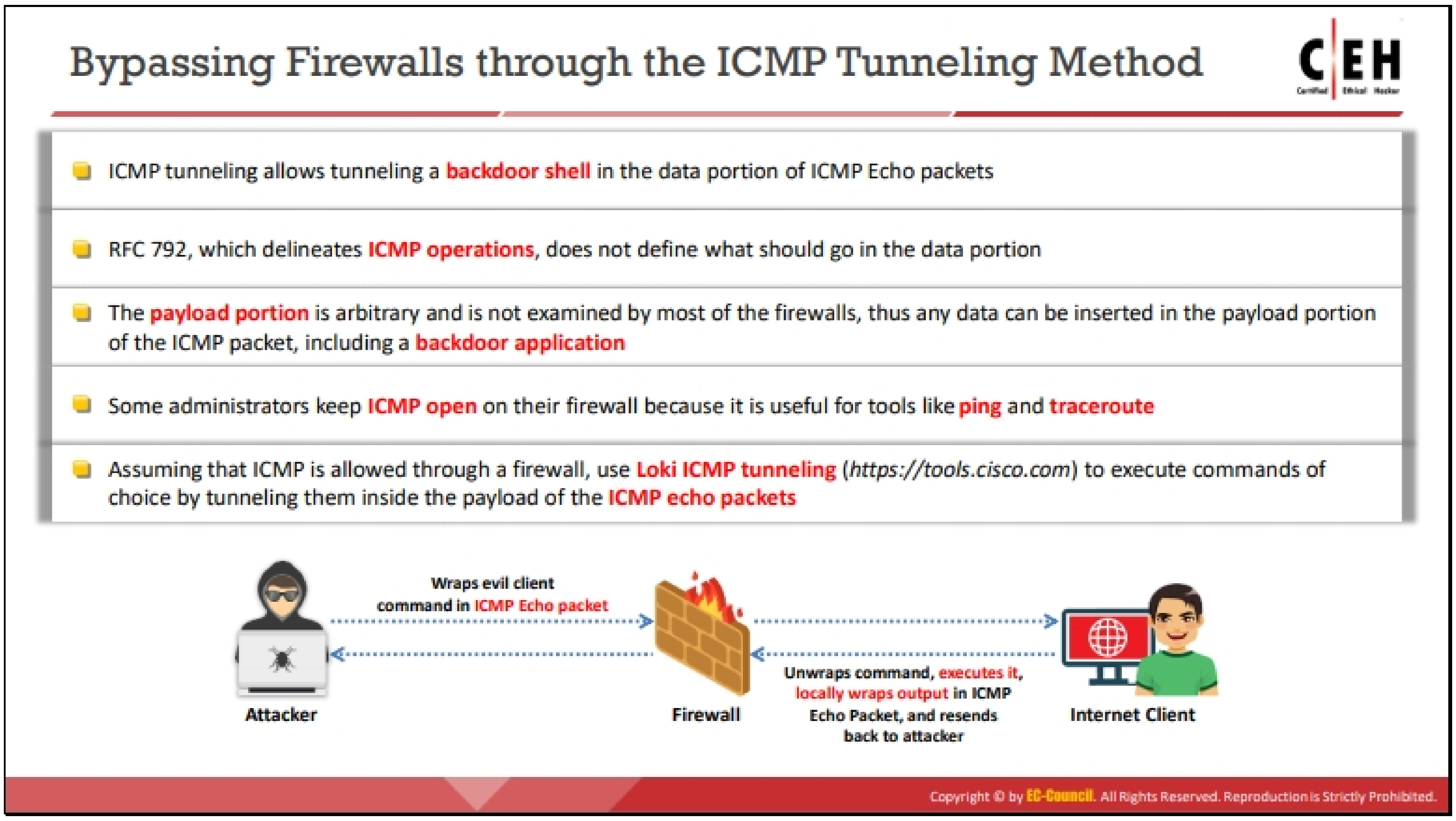
Bypass a Firewall Using a Proxy Server

1. Find an appropriate **proxy server**
2. On the Tools menu of any Internet browser, go to "**Proxy Settings**" and in the **Internet Properties** dialog box under the **Connections** tab, click "**LAN settings**"
3. Under **LAN Settings**, click on the "**Use a proxy server for your LAN**" check box
4. In the **Address** box, type the **IP address** of the proxy server
5. In the **Port** box, type the **port number** that is used by the proxy server for client connections (by default, 8080)
6. Click to select "**Bypass proxy server for local addresses**" check box if you do not want the proxy server computer to be used when connected to a computer on the local network
7. Click **OK** to close the **LAN Settings** dialog box
8. Click **OK** again to close the **Internet Properties** dialog box

Bypass a Firewall Using a Proxy Server

Steps to be followed to bypass a firewall using a proxy server:

1. Find an appropriate proxy server
2. In the Tools menu of any Internet browser, go to "**Proxy Settings,**" and in the **Internet Properties** dialog box under **Connections** tab, click "**LAN settings**"
3. Under **LAN Settings**, click on the "**Use a proxy server for your LAN**" checkbox
4. In the **Address** box, type the **IP address** of the proxy server
5. In the **Port** box, type the **port number** that is used by the proxy server for client connections (by default, 8080)
6. Click to select the "**Bypass proxy server for local addresses**" checkbox if you do not want the proxy server computer to be used when connected to a computer on the local network
7. Click **OK** to close the **LAN Settings** dialog box
8. Click **OK** again to close the **Internet Properties** dialog box



Bypassing Firewalls through the ICMP Tunneling Method

- ICMP tunneling allows tunneling a **backdoor shell** in the data portion of ICMP Echo packets
- RFC 792, which delineates **ICMP operations**, does not define what should go in the data portion
- The **payload portion** is arbitrary and is not examined by most of the firewalls, thus any data can be inserted in the payload portion of the ICMP packet, including a **backdoor application**
- Some administrators keep **ICMP open** on their firewall because it is useful for tools like **ping** and **traceroute**
- Assuming that ICMP is allowed through a firewall, use **Loki ICMP tunneling** (*https://tools.cisco.com*) to execute commands of choice by tunneling them inside the payload of the **ICMP echo packets**

Wraps evil client command in ICMP Echo packet
Attacker
Firewall
Unwraps command, executes it, locally wraps output in ICMP Echo Packet, and resends back to attacker
Internet Client

Bypassing Firewalls through the ICMP Tunneling Method

The ICMP protocol is used to send an error message to the client. As it is a required service for network communication, users often enable this service on their networks. Moreover, it does not entail a significant threat from the security perspective. The attacker takes advantage of the enabled ICMP protocol on the network and performs ICMP tunneling to send his/her malicious data into the target network. The ICMP tunnel provides attackers with full access to target networks.

It allows tunneling of a backdoor shell in the data portion of ICMP Echo packets. RFC 792, which delineates ICMP operation, does not define what should go in the data portion. The payload portion is arbitrary and is not examined by most firewalls. Thus, any data can be inserted in the payload portion of the ICMP packet, including a backdoor application. Some administrators keep ICMP open on their firewall because it is useful for tools such as ping and traceroute. Assuming that ICMP is allowed through a firewall, use Loki ICMP tunneling (*https://tools.cisco.com*) to execute commands of choice by tunneling them inside the payload of ICMP echo packets.

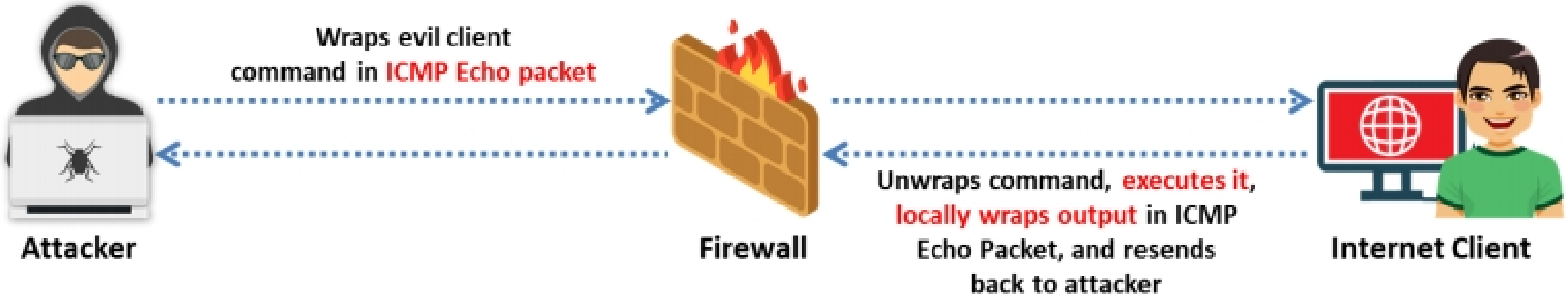
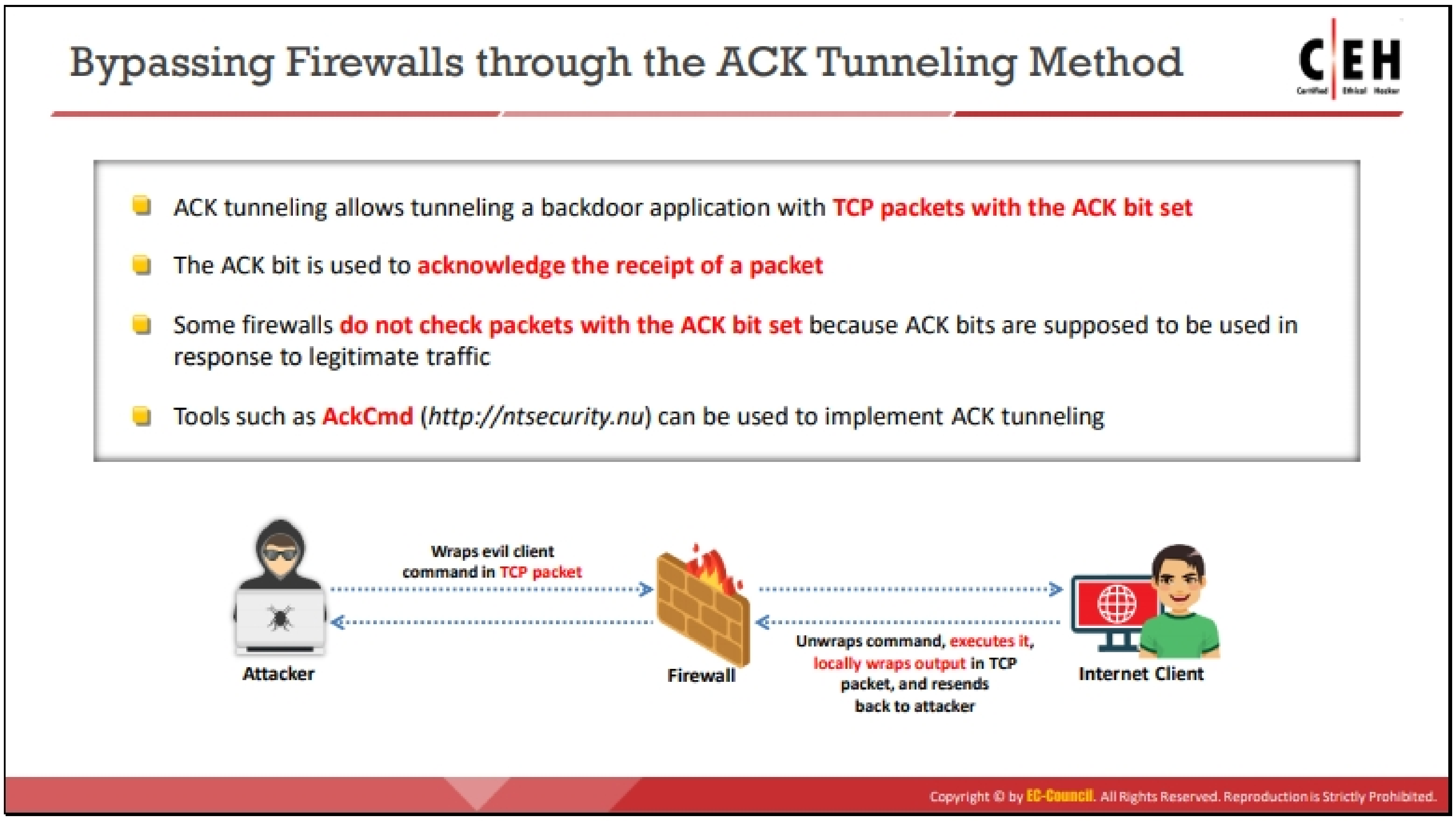


Figure 12.44: Bypassing firewall through ICMP tunneling



Bypassing Firewalls through the ACK Tunneling Method

- ACK tunneling allows tunneling a backdoor application with **TCP packets with the ACK bit set**
- The ACK bit is used to **acknowledge the receipt of a packet**
- Some firewalls **do not check packets with the ACK bit set** because ACK bits are supposed to be used in response to legitimate traffic
- Tools such as **AckCmd** (*http://ntsecurity.nu*) can be used to implement ACK tunneling

Wraps evil client command in TCP packet

Attacker

Firewall

Unwraps command, executes it, locally wraps output in TCP packet, and resends back to attacker

Internet Client

Bypassing Firewalls through the ACK Tunneling method

Ordinary packet filtering firewalls define their rule sets based on the SYN packet when TCP level communication is to be established. This is because such a firewall assumes that only the SYN packet is coming from the client and is thus likely to contain malicious code in the SYN packet. These firewalls ignore the possibility that the attacker can also inject malicious code in the ACK packet. As ACK packets are sent after establishing a session, ACK traffic is considered legitimate. In addition, the filtering of ACK packets is ignored to reduce the workload of firewalls, as there can be many ACK packets for one SYN packet. ACK tunneling allows tunneling of a backdoor application with TCP packets with the ACK bit set. The ACK bit acknowledges the receipt of a packet. As stated earlier, some firewalls do not check packets with the ACK bit set, because ACK bits are supposed to be used in response to legitimate traffic that has already been allowed to pass through. Attackers exploit this fact in ACK tunneling. Tools such as **AckCmd** (*http://ntsecurity.nu*) use ACK tunneling.

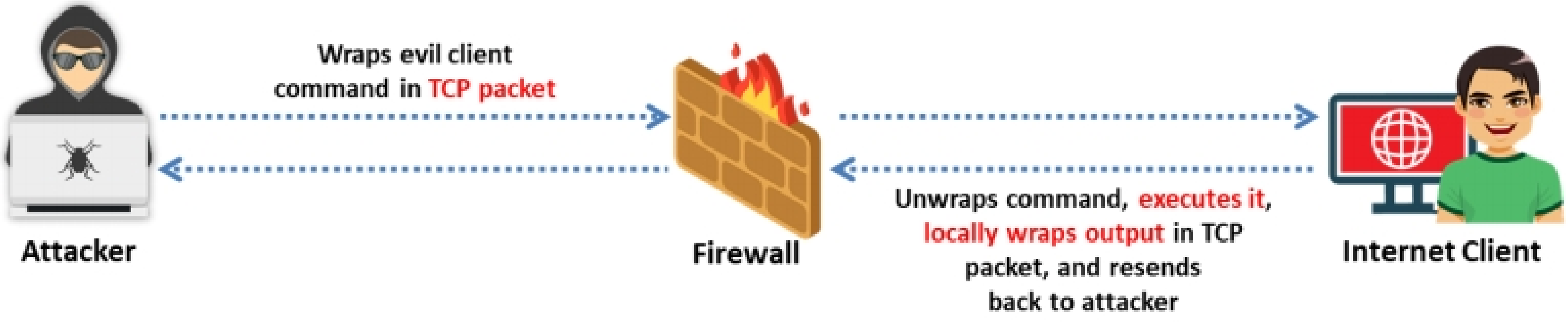
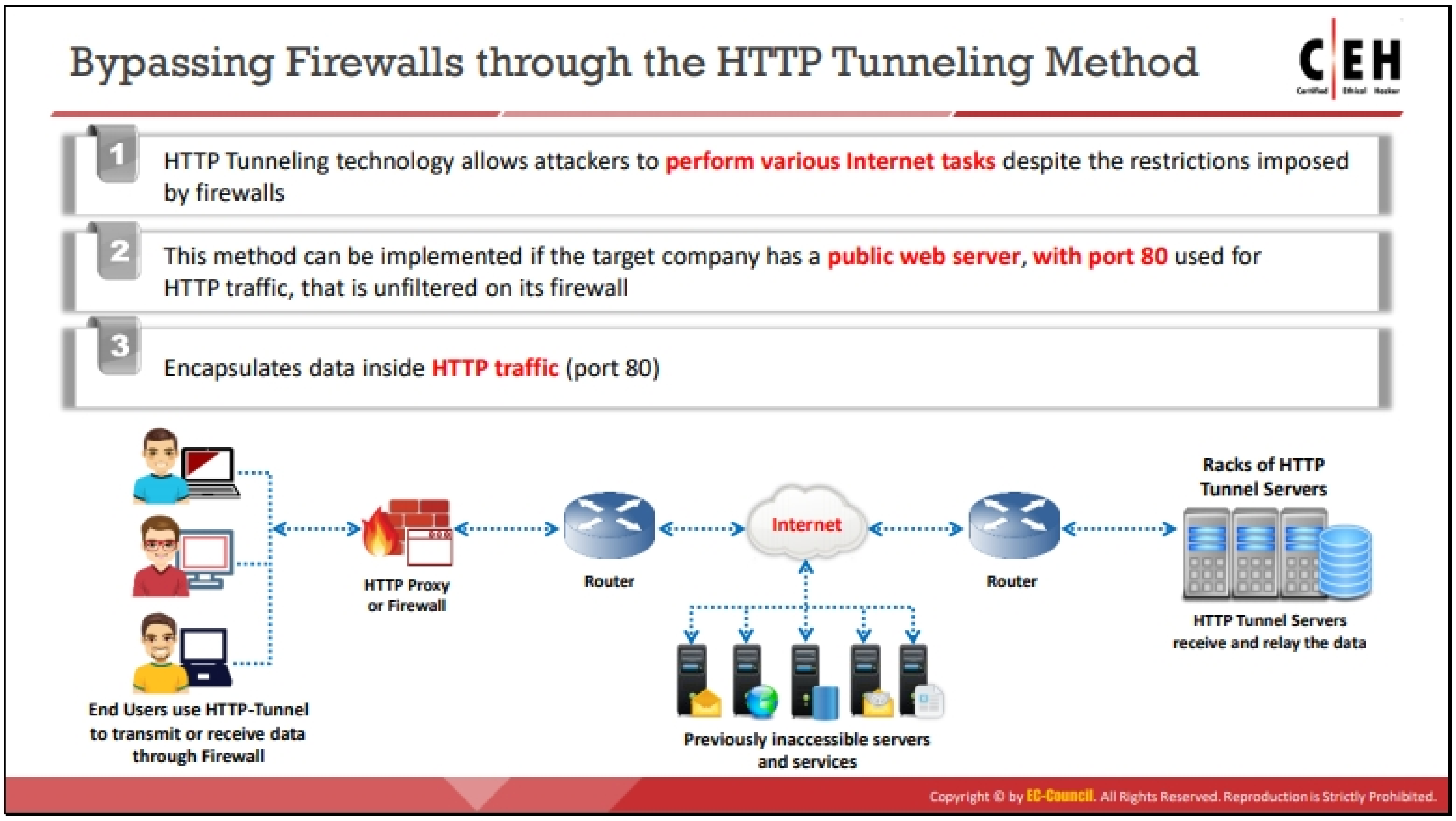


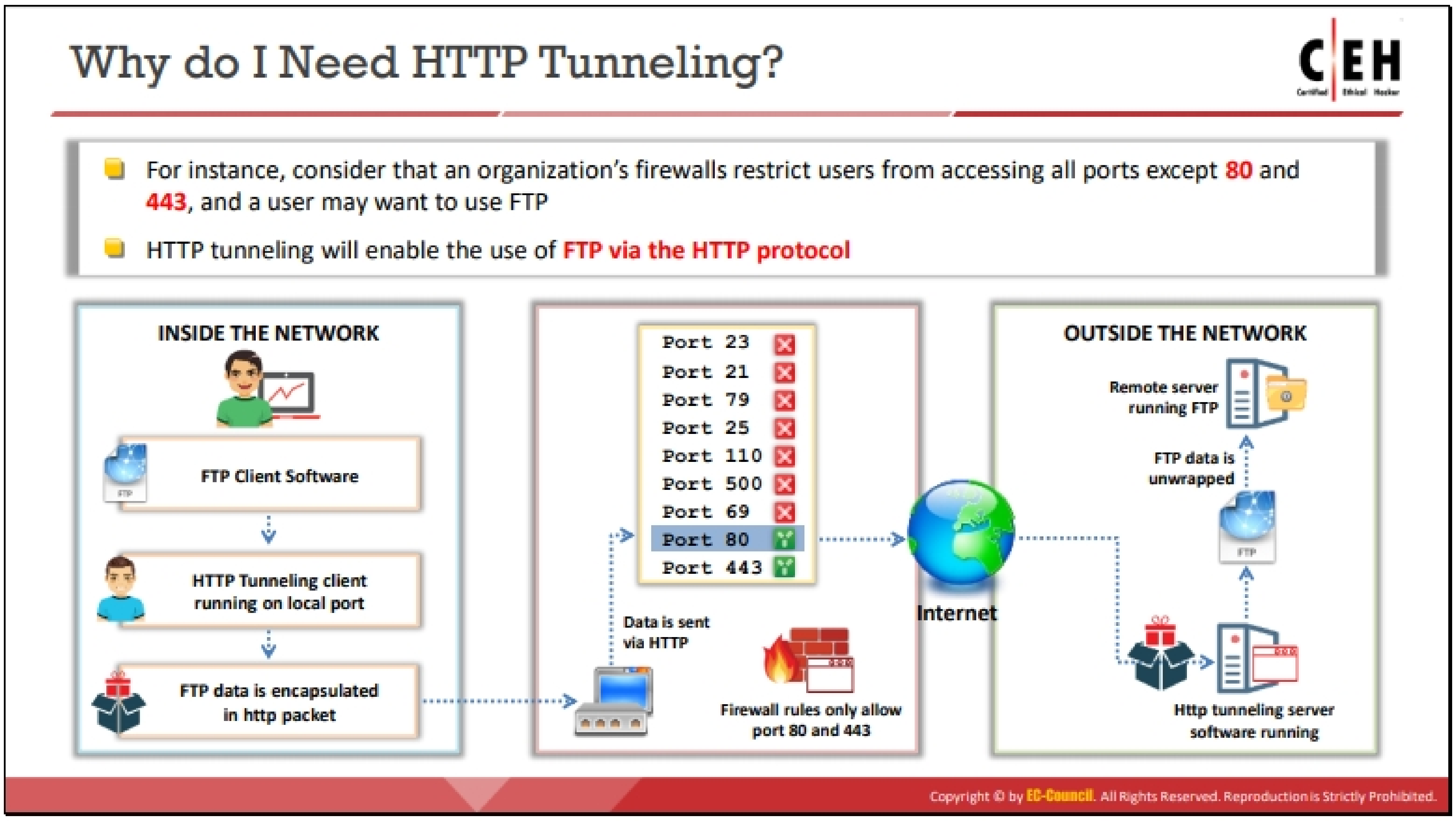
Figure 12.45: Bypassing firewall through ACK tunneling



Bypassing Firewalls through the HTTP Tunneling Method

1. HTTP Tunneling technology allows attackers to **perform various Internet tasks** despite the restrictions imposed by firewalls
2. This method can be implemented if the target company has a **public web server**, **with port 80** used for HTTP traffic, that is unfiltered on its firewall
3. Encapsulates data inside **HTTP traffic** (port 80)

End Users use HTTP-Tunnel to transmit or receive data through Firewall
HTTP Proxy or Firewall
Router
Internet
Router
Racks of HTTP Tunnel Servers
HTTP Tunnel Servers receive and relay the data
Previously inaccessible servers and services



Why do I Need HTTP Tunneling?

- For instance, consider that an organization's firewalls restrict users from accessing all ports except **80** and **443**, and a user may want to use FTP
- HTTP tunneling will enable the use of **FTP via the HTTP protocol**

INSIDE THE NETWORK
FTP Client Software
HTTP Tunneling client running on local port
FTP data is encapsulated in http packet
Port 23 ☒
Port 21 ☒
Port 79 ☒
Port 25 ☒
Port 110 ☒
Port 500 ☒
Port 69 ☒
Port 80 ✓
Port 443 ✓
Data is sent via HTTP
Firewall rules only allow port 80 and 443
Internet
OUTSIDE THE NETWORK
Remote server running FTP
FTP data is unwrapped
FTP
Http tunneling server software running

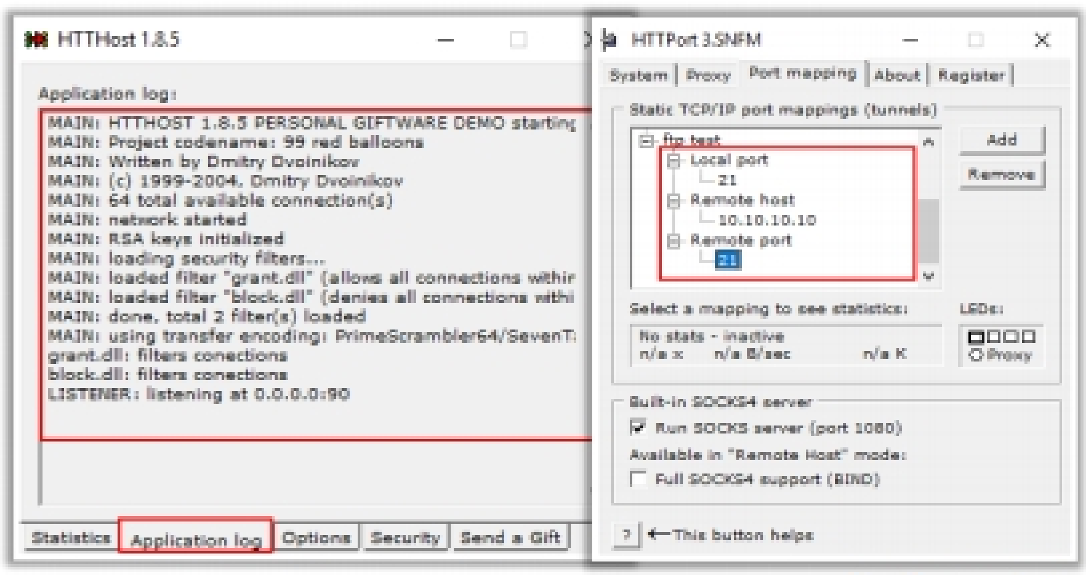
HTTP Tunneling Tools

HTTPort and HTTHost

- HTTPort allows you to **bypass your HTTP proxy**, which is blocking you from the Internet
- It allows you to use various **Internet software from behind the proxy**, such as e-mail, instant messengers, P2P file sharing, ICQ, News, FTP, or IRC

Super Network Tunnel

- A **two-way http tunnel** software connecting two computers
- It works like **VPN tunneling** but uses HTTP protocol to establish a connection


HTTHost 1.8.5
Application log:
Statistics
Application log
Options
Security
Send a Gift
HTTPort 3.SNFM
System
Proxy
Port mapping
About
Register
Static TCP/IP port mappings (tunnels)
Add
Remove
Select a mapping to see statistics:
LEDs:
Proxy
Built-in SOCKS4 server
Available in "Remote Host" mode:
This button helps

```
MAIN: HTTHOST 1.8.5 PERSONAL GIFTWARE DEMO startinç
MAIN: Project codename: 99 red balloons
MAIN: Written by Dmitry Dvoinikov
MAIN: (c) 1999-2004, Dmitry Dvoinikov
MAIN: 64 total available connection(s)
MAIN: network started
MAIN: RSA keys initialized
MAIN: loading security filters...
MAIN: loaded filter "grant.dll" (allows all connections withir
MAIN: loaded filter "block.dll" (denies all connections withi
MAIN: done, total 2 filter(s) loaded
MAIN: using transfer encoding: PrimeScrambler64/SevenT:
grant.dll: filters conections
block.dll: filters conections
LISTENER: listening at 0.0.0.0:90
```

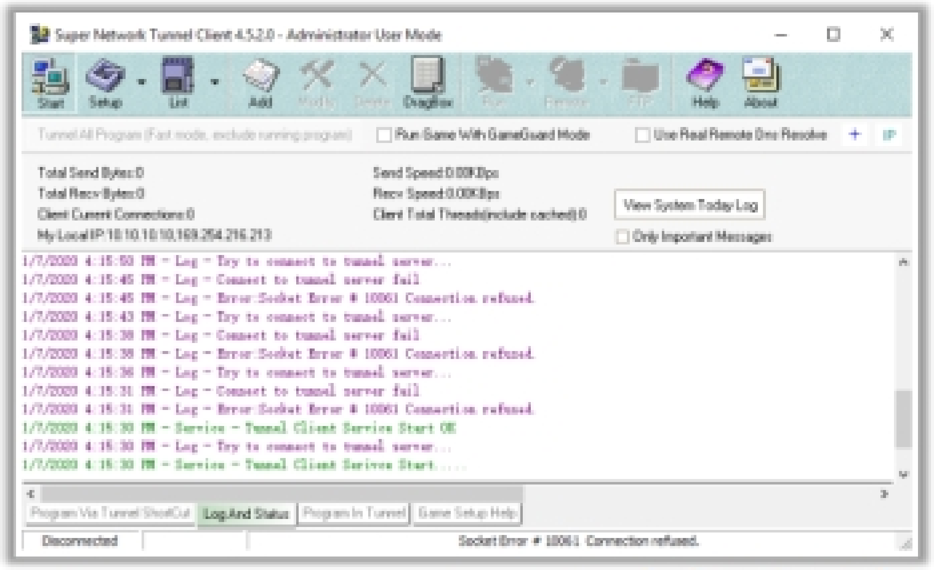
```
ftp test
  Local port
    21
  Remote host
    10.10.10.10
  Remote port
    21
```

No stats - inactive
n/a x   n/a B/sec   n/a K

☑ Run SOCKS server (port 1080)

☐ Full SOCKS4 support (BIND)

https://www.targeted.org


Super Network Tunnel Client 4.5.2.0 - Administrator User Mode
Start
Setup
List
Add
Modify
Delete
DiagBox
Run
Remote
FTP
Help
About
Tunnel All Program (Fast mode, exclude running program)
View System Today Log
Program Via Tunnel ShortCut
Log And Status
Program In Tunnel
Game Setup Help

☐ Run Game With GameGuard Mode
☐ Use Real Remote Dns Resolve

Total Send Bytes:0
Total Recv Bytes:0
Client Current Connections:0
My Local IP:10.10.10.10,169.254.216.213
Send Speed:0.00KBps
Recv Speed:0.00KBps
Client Total Threads(include cached):0

☐ Only Important Messages

```
1/7/2020 4:15:50 PM - Log - Try to connect to tunnel server...
1/7/2020 4:15:46 PM - Log - Connect to tunnel server fail
1/7/2020 4:15:46 PM - Log - Error:Socket Error # 10061 Connection refused.
1/7/2020 4:15:43 PM - Log - Try to connect to tunnel server...
1/7/2020 4:15:38 PM - Log - Connect to tunnel server fail
1/7/2020 4:15:38 PM - Log - Error:Socket Error # 10061 Connection refused.
1/7/2020 4:15:36 PM - Log - Try to connect to tunnel server...
1/7/2020 4:15:31 PM - Log - Connect to tunnel server fail
1/7/2020 4:15:31 PM - Log - Error:Socket Error # 10061 Connection refused.
1/7/2020 4:15:30 PM - Service - Tunnel Client Service Start OK
1/7/2020 4:15:30 PM - Log - Try to connect to tunnel server...
1/7/2020 4:15:30 PM - Service - Tunnel Client Serivce Start.....
```

Disconnected   Socket Error # 10061 Connection refused.

http://www.networktunnel.net

Bypassing Firewalls through the HTTP Tunneling Method

HTTP tunneling allows attackers to perform various Internet tasks despite the restrictions imposed by firewalls. This method can be implemented if the target company has a public web server in which port 80 is used for HTTP traffic that is unfiltered by its firewall. The attacker encapsulates data inside HTTP traffic (via port 80). Many firewalls do not examine the payload of an HTTP packet to confirm that it is legitimate. Thus, it is possible to tunnel traffic via TCP port 80.

Tools such as **HTTPTunnel** (*http://http-tunnel.sourceforge.net*) use this technique of tunneling traffic across TCP port 80. HTTPTunnel is a client/server application, the client application is htc, and the server is hts. Upload the server to the target system and redirect it through TCP port 80.

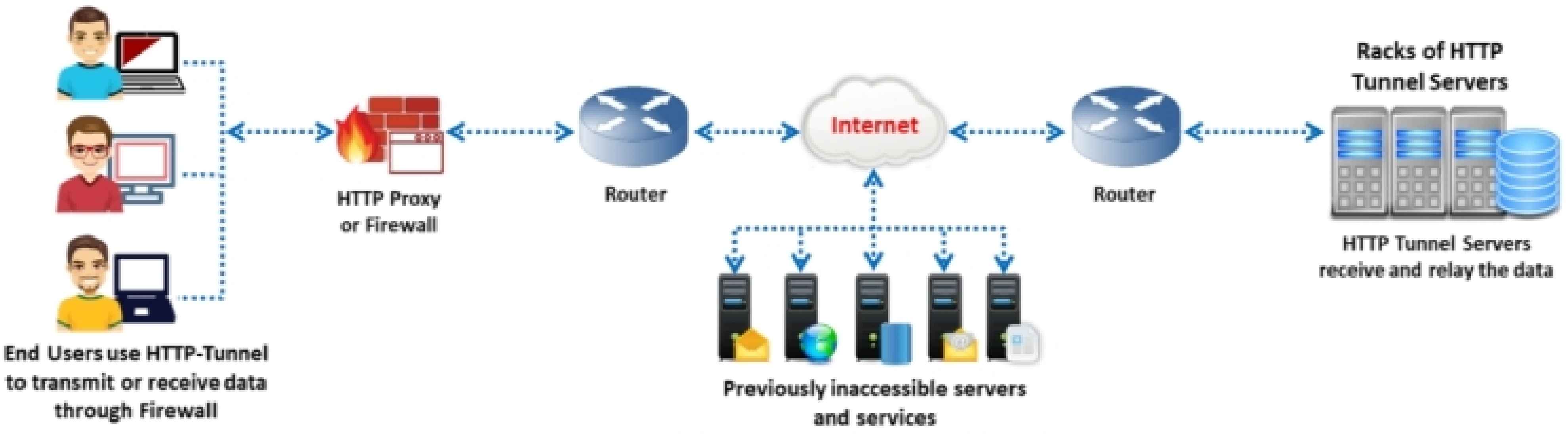

End Users use HTTP-Tunnel to transmit or receive data through Firewall
HTTP Proxy or Firewall
Router
Internet
Router
Racks of HTTP Tunnel Servers
HTTP Tunnel Servers receive and relay the data
Previously inaccessible servers and services

Figure 12.46: Bypassing firewall through HTTP tunneling

Why do I Need HTTP Tunneling?

HTTP tunneling is used in scenarios in which network users are granted restricted connectivity through a firewall or proxy; in such conditions, some applications may also lack native communications support.

These restrictions include:

- Blocking of TCP/IP ports, traffic initiated from outside the network, and network protocols except for a few commonly used protocols, etc.
- Surfing blocked websites
- Posting in forums anonymously by hiding the IP address
- Using an application such as chatting through ICQ or IRC, instant messengers, games, browsers, etc.
- Sharing of confidential resources over HTTP securely
- Downloading files with filtered extensions and/or with malicious code

For instance, consider that organization firewalls restrict users to access all ports except 80 and 443, and a user may want to use FTP. HTTP tunneling enables FTP use via the HTTP protocol. The HTTP tunnel creates a bidirectional virtual data connection tunneled in HTTP traffic. It works with the help of FTP client software to perform protocol encapsulation by enclosing data packets of one protocol such as SOAP or JRMP within HTTP packets on, e.g., local port 80. These packets are sent through the firewall or proxy server as normal Internet traffic, which is then directed to the HTTP tunneling server software located outside the network. Upon receiving the packets, this server unwraps the FTP data and redirects the packet to the remote FTP server.

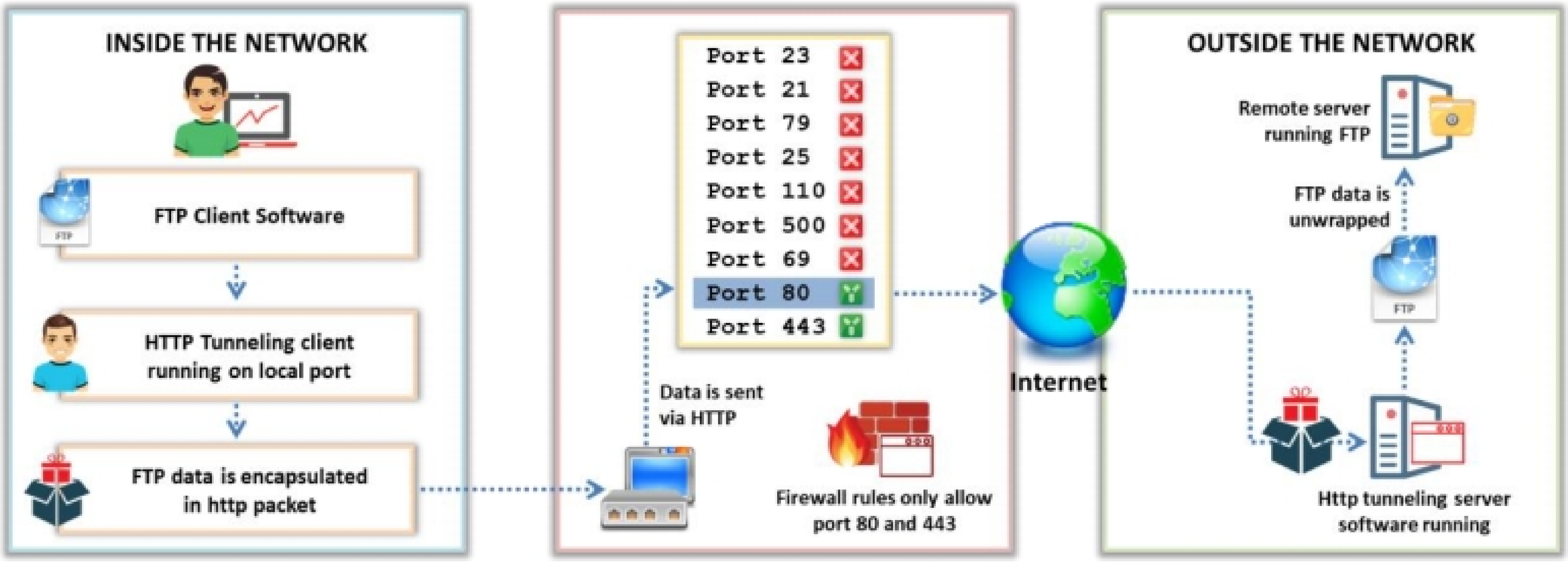

INSIDE THE NETWORK
FTP Client Software
HTTP Tunneling client running on local port
FTP data is encapsulated in http packet
Port 23 ☒
Port 21 ☒
Port 79 ☒
Port 25 ☒
Port 110 ☒
Port 500 ☒
Port 69 ☒
Port 80 ✓
Port 443 ✓
Data is sent via HTTP
Firewall rules only allow port 80 and 443
Internet
OUTSIDE THE NETWORK
Remote server running FTP
FTP data is unwrapped
Http tunneling server software running

Figure 12.47: Illustration of HTTP Tunneling Method

HTTP Tunneling Tools

Some HTTP tunneling tools are as follows:

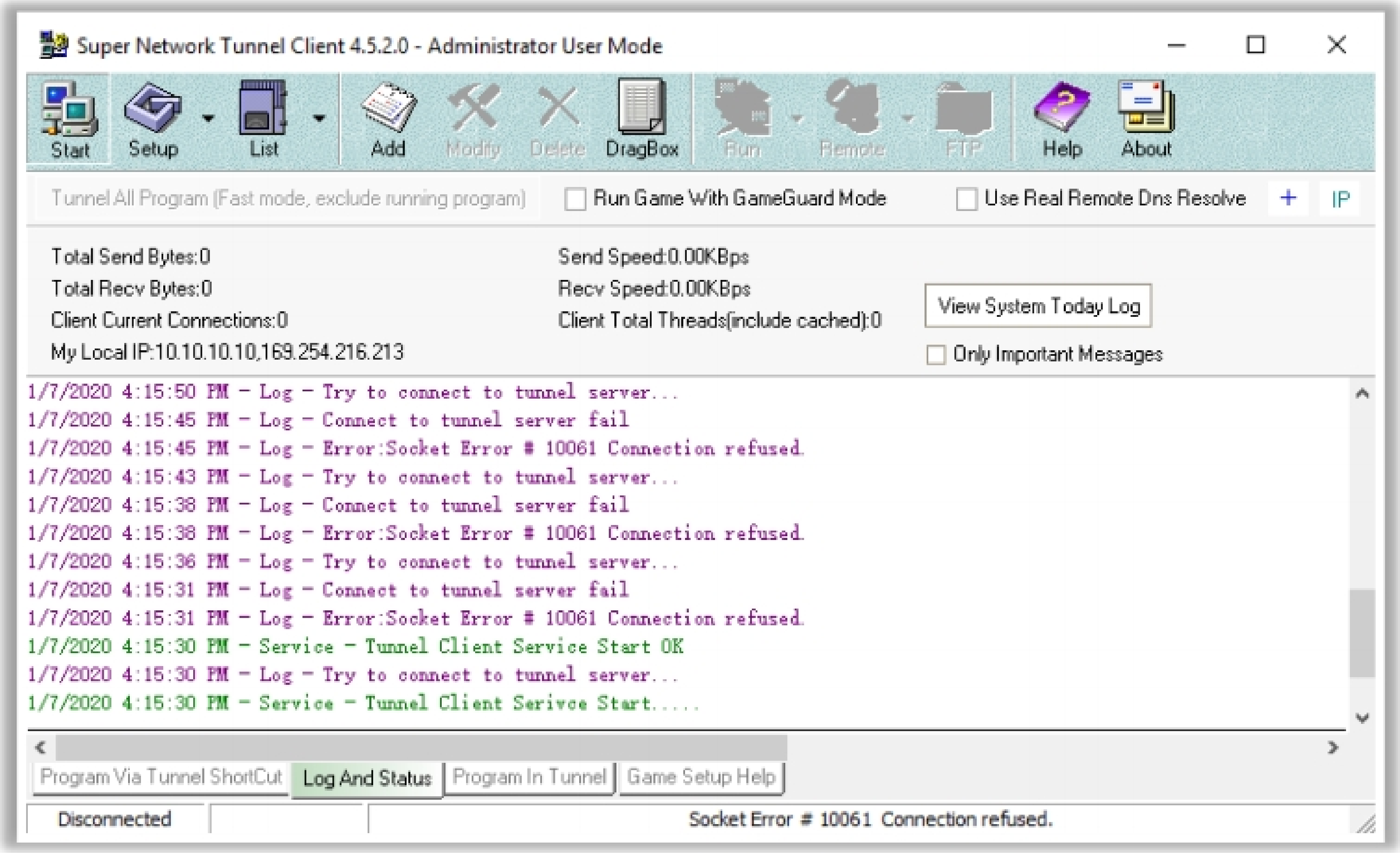
- **Super Network Tunnel**

  Source: *http://www.networktunnel.net*

  Super Network Tunnel is a two-way HTTP tunneling software that connects two computers using HTTP-Tunnel Client and HTTP-Tunnel Server. It works like VPN tunneling but uses the HTTP protocol to establish a connection for accessing the Internet without monitoring and provides an extra layer of protection against attackers, spyware, identity theft, and so on. It can bypass any firewall to surf the web, use IM applications, games, and so on. Further, it integrates the SocksCap function along with bidirectional HTTP tunneling and remote control to simplify the configuration.

This tool allows HTTP, HTTPS, and SOCKS tunneling of any TCP communication between any client-server systems. The TCP traffic is sent from the client to the server via standard HTTP POST requests, which allows penetrating through firewalls, proxy servers, and so on, where HTTP traffic passes.

The client side of a tunnel is the Super Network Tunnel client app, which listens on a particular TCP port for incoming requests. Once the request comes, the program creates an HTTP/HTTPS tunnel to the server and sends data through it. The server side is a Super Network Tunnel server, which simply forwards the data to the intended recipient app running on the server computer or LAN. Both client and server sides support multiple tunnels and multiple connections through the same tunnel at the same time.


Super Network Tunnel Client 4.5.2.0 - Administrator User Mode
Start
Setup
List
Add
Modify
Delete
DragBox
Run
Remote
FTP
Help
About
Tunnel All Program (Fast mode, exclude running program)
☐ Run Game With GameGuard Mode
☐ Use Real Remote Dns Resolve
+
IP
View System Today Log
☐ Only Important Messages
Program Via Tunnel ShortCut
Log And Status
Program In Tunnel
Game Setup Help

Total Send Bytes:0
Total Recv Bytes:0
Client Current Connections:0
My Local IP:10.10.10.10,169.254.216.213
Send Speed:0.00KBps
Recv Speed:0.00KBps
Client Total Threads(include cached):0

```
1/7/2020 4:15:50 PM - Log - Try to connect to tunnel server...
1/7/2020 4:15:45 PM - Log - Connect to tunnel server fail
1/7/2020 4:15:45 PM - Log - Error:Socket Error # 10061 Connection refused.
1/7/2020 4:15:43 PM - Log - Try to connect to tunnel server...
1/7/2020 4:15:38 PM - Log - Connect to tunnel server fail
1/7/2020 4:15:38 PM - Log - Error:Socket Error # 10061 Connection refused.
1/7/2020 4:15:36 PM - Log - Try to connect to tunnel server...
1/7/2020 4:15:31 PM - Log - Connect to tunnel server fail
1/7/2020 4:15:31 PM - Log - Error:Socket Error # 10061 Connection refused.
1/7/2020 4:15:30 PM - Service - Tunnel Client Service Start OK
1/7/2020 4:15:30 PM - Log - Try to connect to tunnel server...
1/7/2020 4:15:30 PM - Service - Tunnel Client Serivce Start.....
```

Disconnected
Socket Error # 10061 Connection refused.

Figure 12.48: Screenshot of Super Network Tunnel

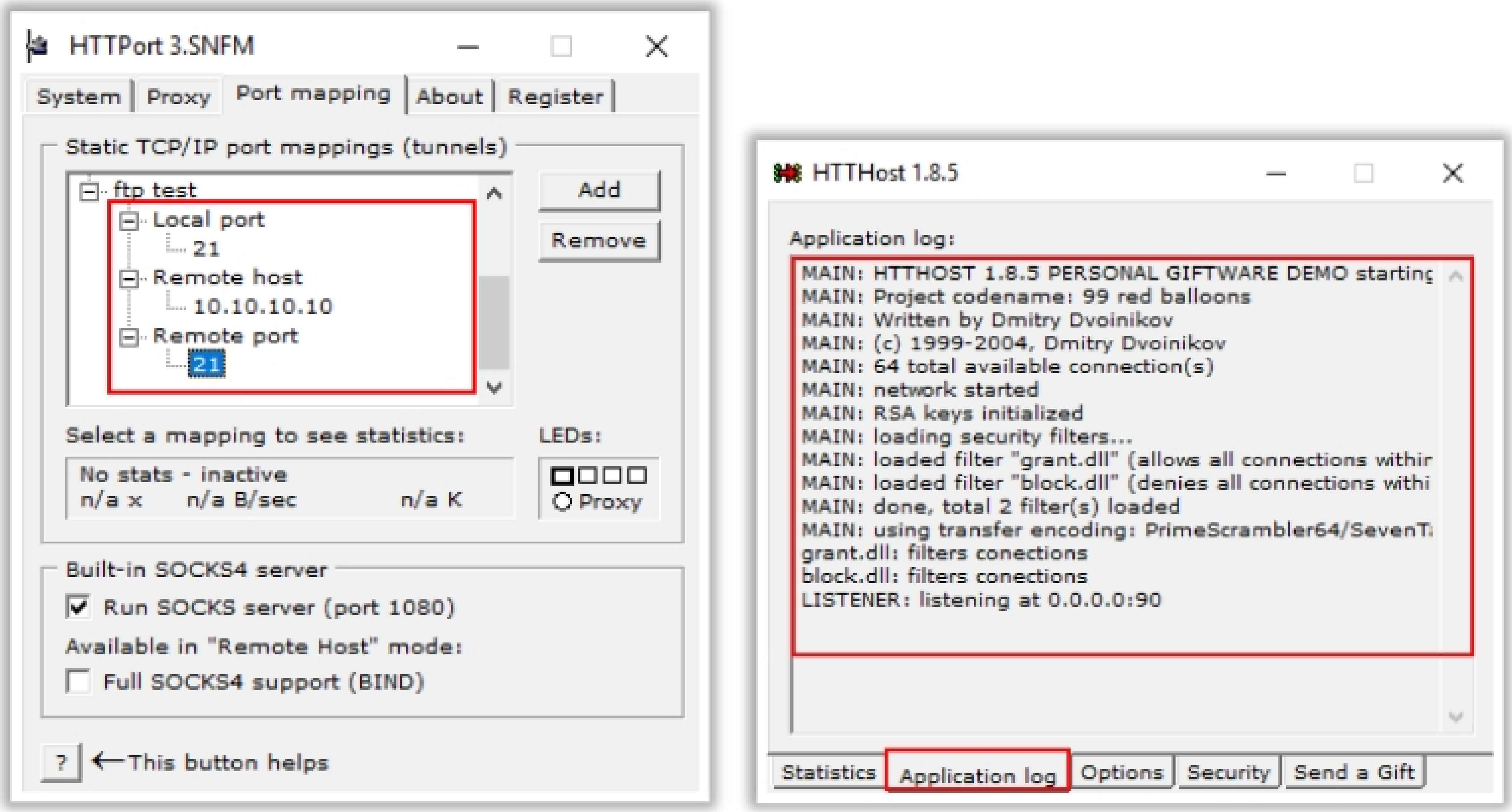
- **HTTPort and HTTHost**

  Source: *https://www.targeted.org*

  HTTPort allows users to bypass the HTTP proxy, which blocks Internet access to e-mail, instant messengers, P2P file sharing, ICQ, News, FTP, IRC, and so on. Here, the Internet software is configured so that it connects to a local PC as if it is the required remote server. HTTPort then intercepts that connection and runs it through a tunnel through the proxy. HTTPort can work on devices such as proxies or firewalls that allow HTTP traffic. Thus, HTTPort provides access to websites and Internet apps. HTTPort performs tunneling using one of two modes: SSL/CONNECT mode or a remote host.

In the SSL/CONNECT mode, HTTPort can make a tunnel through a proxy all by itself. It requires that the proxy should support a particular HTTP feature, specifically, CONNECT HTTP. Most proxies have this method disabled by default. The SSL/CONNECT mode is much faster, but in this case, encryption cannot be used, and the proxy can track all actions.

The remote host method is capable of tunneling through any proxy. HTTPort uses a special server software called HTTHost, which is installed outside the proxy-blocked network. It is a web server; thus, when HTTPort is tunneling, it sends a series of HTTP requests to the HTTHost. The proxy responds as if the user is surfing a website and thus allows the user to do so. HTTHost, in turn, performs its half of the tunneling and communicates with the target servers. This mode is much slower but works in most cases and features strong data encryption that makes proxy logging useless.



Static TCP/IP port mappings (tunnels)

- ftp test
  - Local port
    - 21
  - Remote host
    - 10.10.10.10
  - Remote port
    - 21

Select a mapping to see statistics:

No stats - inactive
n/a x n/a B/sec n/a K

Built-in SOCKS4 server
☑ Run SOCKS server (port 1080)
Available in "Remote Host" mode:
☐ Full SOCKS4 support (BIND)

Application log:

```
MAIN: HTTHOST 1.8.5 PERSONAL GIFTWARE DEMO startin
MAIN: Project codename: 99 red balloons
MAIN: Written by Dmitry Dvoinikov
MAIN: (c) 1999-2004, Dmitry Dvoinikov
MAIN: 64 total available connection(s)
MAIN: network started
MAIN: RSA keys initialized
MAIN: loading security filters...
MAIN: loaded filter "grant.dll" (allows all connections withi
MAIN: loaded filter "block.dll" (denies all connections withi
MAIN: done, total 2 filter(s) loaded
MAIN: using transfer encoding: PrimeScrambler64/SevenT
grant.dll: filters conections
block.dll: filters conections
LISTENER: listening at 0.0.0.0:90
```

Figure 12.49: Screenshot of HTTPort and HTTHost

- **Other HTTP Tunneling Tools**
  - Tunna (*https://github.com*)
  - HTTP Tunnel (*http://http-tunnel.sourceforge.net*)

Bypassing Firewalls through the SSH Tunneling Method

OpenSSH

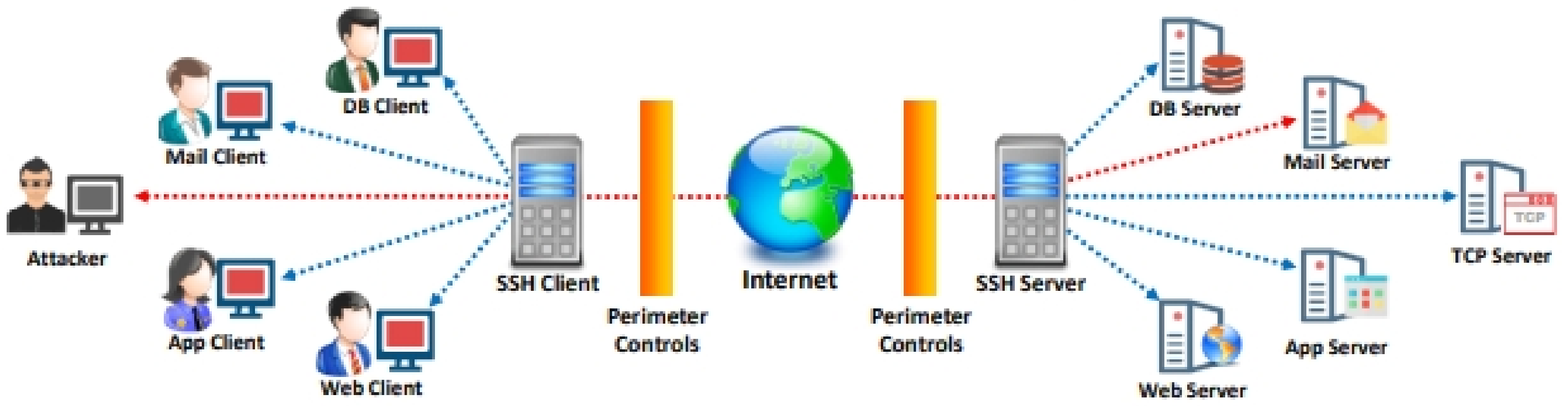
- Attackers use OpenSSH to **encrypt and tunnel all the traffic** from a local machine to a remote machine to avoid detection by the perimeter security controls

Example

`ssh -f user@certifiedhacker.com -L 5000:certifiedhacker.com:25 -N`

`-f` => background mode, `user@certifiedhacker.com` => username and server you are logging into, `-L 5000:certifiedhacker.com:25` => local-port:host:remote-port, and `-N` => Do not execute the command on the remote system

- This forwards the **local port 5000 to port 25** on certifiedhacker.com encrypted
- Simply point your email client to use localhost:5000 as the SMTP server


DB Client
Mail Client
Attacker
App Client
Web Client
SSH Client
Perimeter Controls
Internet
Perimeter Controls
SSH Server
DB Server
Mail Server
TCP Server
App Server
Web Server

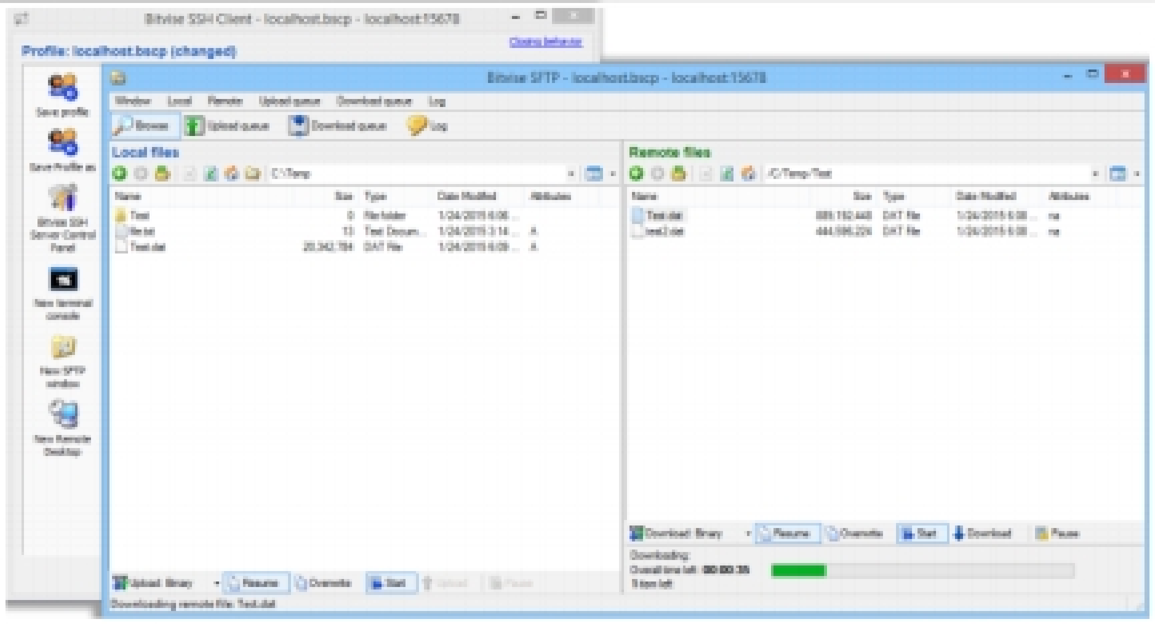
SSH Tunneling Tools: Bitvise and Secure Pipes

Bitvise

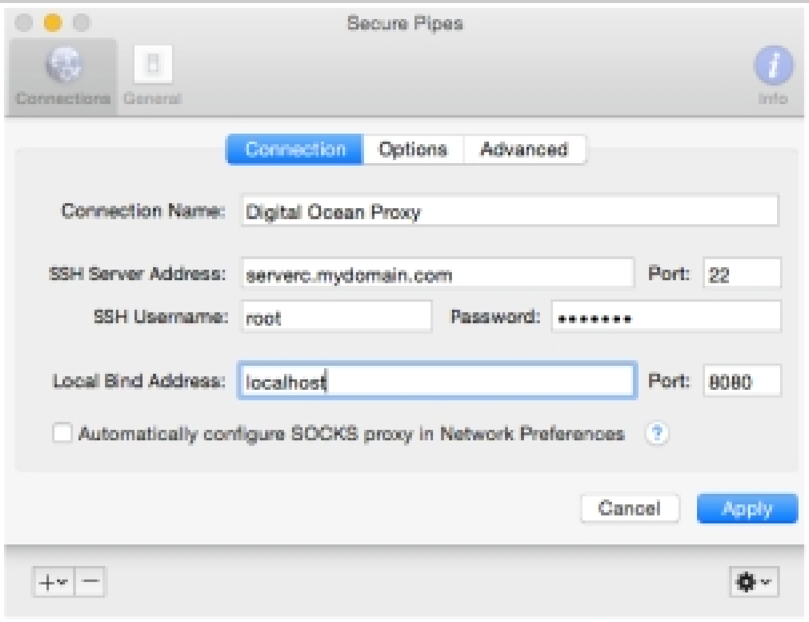
- Bitvise SSH Server provides secure **remote login capabilities** for Windows workstations and servers
- SSH Client includes powerful tunnelling features including **dynamic port forwarding** through an integrated proxy and **remote administration** for the SSH Server

Secure Pipes

- Secure Pipes makes **managing SSH tunnels** simple
- It selectively **opens access** to application ports normally not easily accessible due to network or service provider configuration restrictions


Bitvise SSH Client - localhost.bscp - localhost:15678
Profile: localhost.bscp (changed)
Bitvise SFTP - localhost.bscp - localhost:15678
Local files
Remote files

https://www.bitvise.com


Secure Pipes
Connections
General
Info
Connection
Options
Advanced
Cancel
Apply

Connection Name: Digital Ocean Proxy
SSH Server Address: serverc.mydomain.com Port: 22
SSH Username: root Password: •••••••
Local Bind Address: localhost Port: 8080
☐ Automatically configure SOCKS proxy in Network Preferences

https://www.opoet.com

Bypassing Firewalls through the SSH Tunneling Method

SSH protocol tunneling involves sending unencrypted network traffic through an SSH tunnel. For example, suppose you want to transfer files on an unencrypted FTP protocol, but the FTP protocol is blocked on the target firewall. The unencrypted data can be sent over the encrypted SSH protocol using SSH tunneling. Attackers use this technique to bypass firewall restrictions. They connect to external SSH servers and create SSH tunnels to port 80 on the remote server, thereby bypassing firewall restrictions.

Attackers use OpenSSH (OpenBSD Secure Shell) to encrypt and tunnel all traffic from a local machine to a remote machine to avoid detection by perimeter security controls. OpenSSH is a set of computer programs that provide encrypted communication sessions over a computer network using the SSH protocol.

Example:

```
ssh -f user@certifiedhacker.com -L 5000:certifiedhacker.com:25 -N
```

`-f` => background mode, **`user@certifiedhacker.com`** => username and server you are logging into, **`-L 5000:certifiedhacker.com:25`** => local-port:host:remote-port, and **`-N`** => Do not execute the command on the remote system.

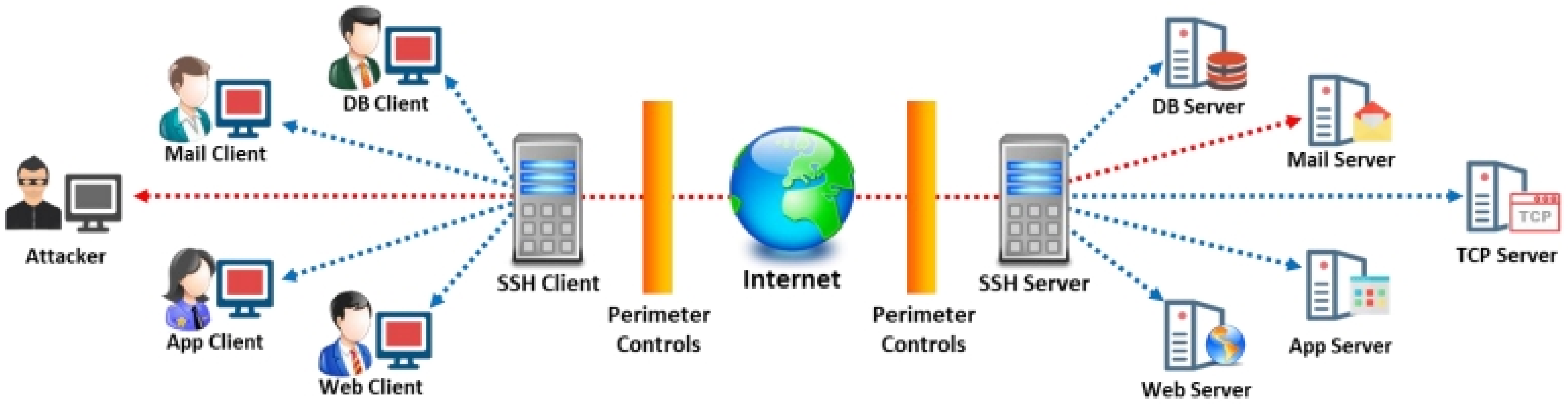


Figure 12.50: Bypassing Firewall through SSH Tunneling Method

SSH Tunneling Tools

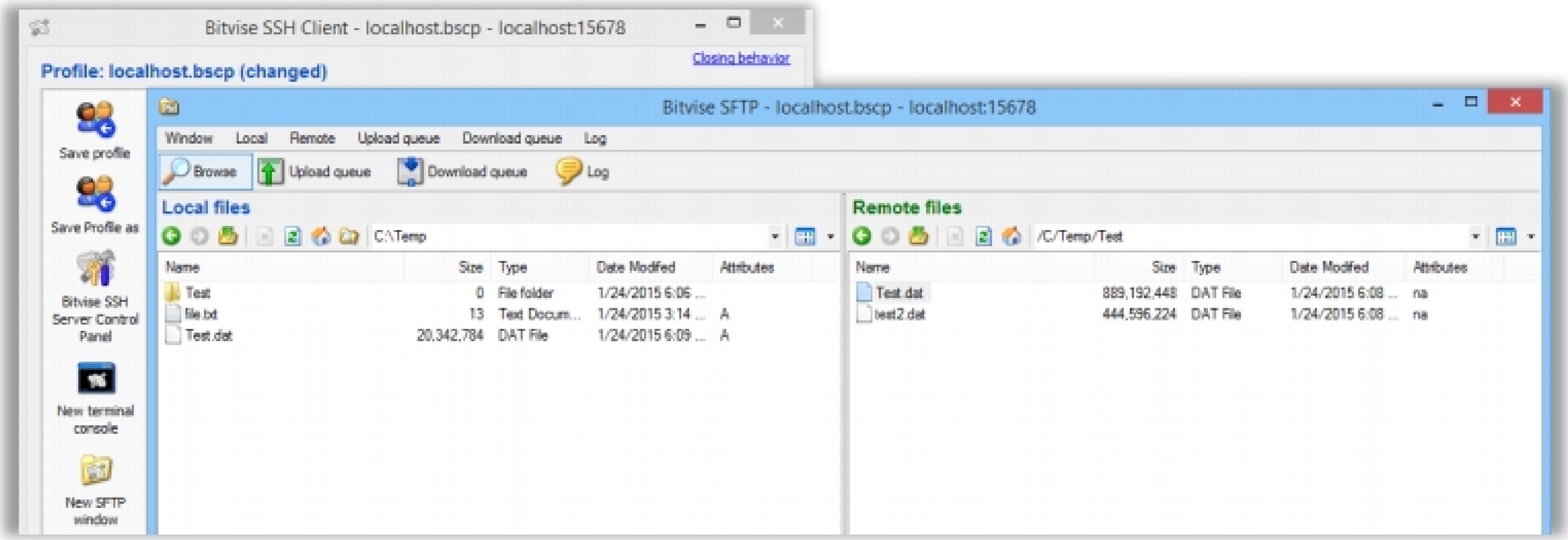
Some SSH tunneling tools are listed below:

- **Bitvise**

  Source: *https://www.bitvise.com*

  Bitvise SSH Server provides secure remote login capabilities to Windows workstations and servers by encrypting data during transmission. It is ideal for remote administration of Windows servers, for advanced users who wish to access their home machine from work or their work machine from home, and for a wide spectrum of advanced tasks, such as establishing a VPN using the SSH TCP/IP tunneling feature or providing a secure file depository using SFTP.

  Bitvise SSH Client for Windows includes terminal emulation, graphical as well as command-line SFTP support, an FTP-to-SFTP bridge, tunneling features—including dynamic port forwarding through an integrated proxy—and remote administration for SSH Server.


Bitvise SSH Client - localhost.bscp - localhost:15678
Profile: localhost.bscp (changed)
Closing behavior
Save profile
Save Profile as
Bitvise SSH Server Control Panel
New terminal console
New SFTP window
Bitvise SFTP - localhost.bscp - localhost:15678
Window
Local
Remote
Upload queue
Download queue
Log
Browse
Upload queue
Download queue
Log
Local files
Remote files

Local files: C:\Temp

Name	Size	Type	Date Modifed	Attributes
Test	0	File folder	1/24/2015 6:06 ...	
file.txt	13	Text Docum...	1/24/2015 3:14 ...	A
Test.dat	20,342,784	DAT File	1/24/2015 6:09 ...	A

Remote files: /C/Temp/Test

Name	Size	Type	Date Modifed	Attributes
Test.dat	889,192,448	DAT File	1/24/2015 6:08 ...	na
test2.dat	444,596,224	DAT File	1/24/2015 6:08 ...	na

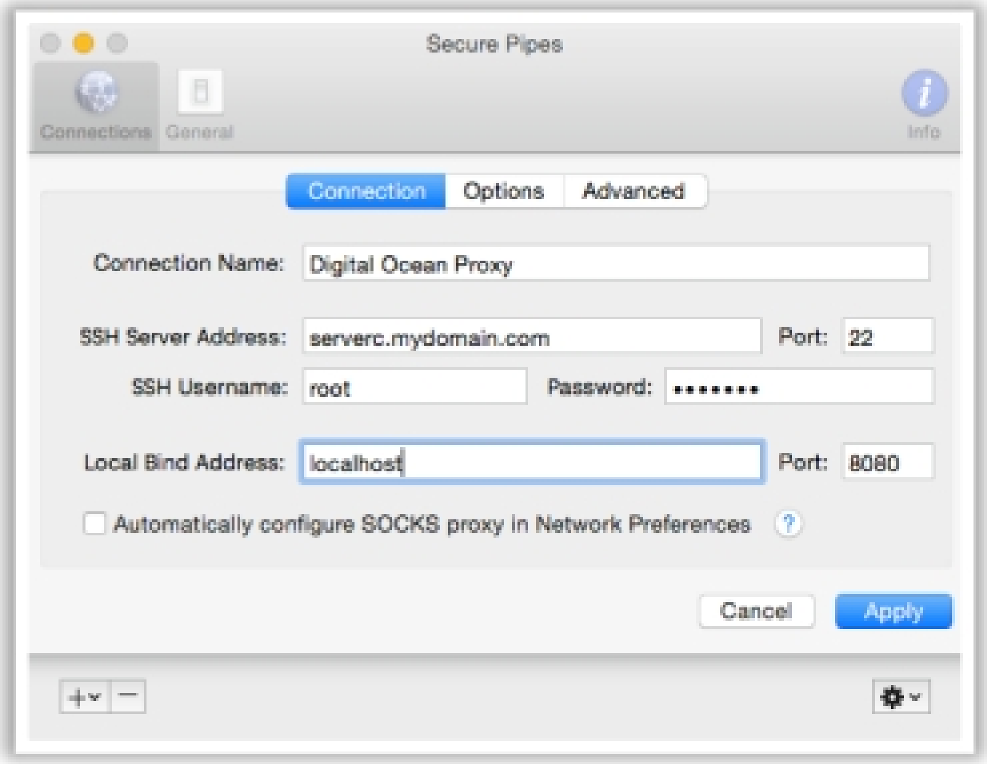
Figure 12.51: Screenshot of Bitvise

- **Secure Pipes**

  Source: *https://www.opoet.com*

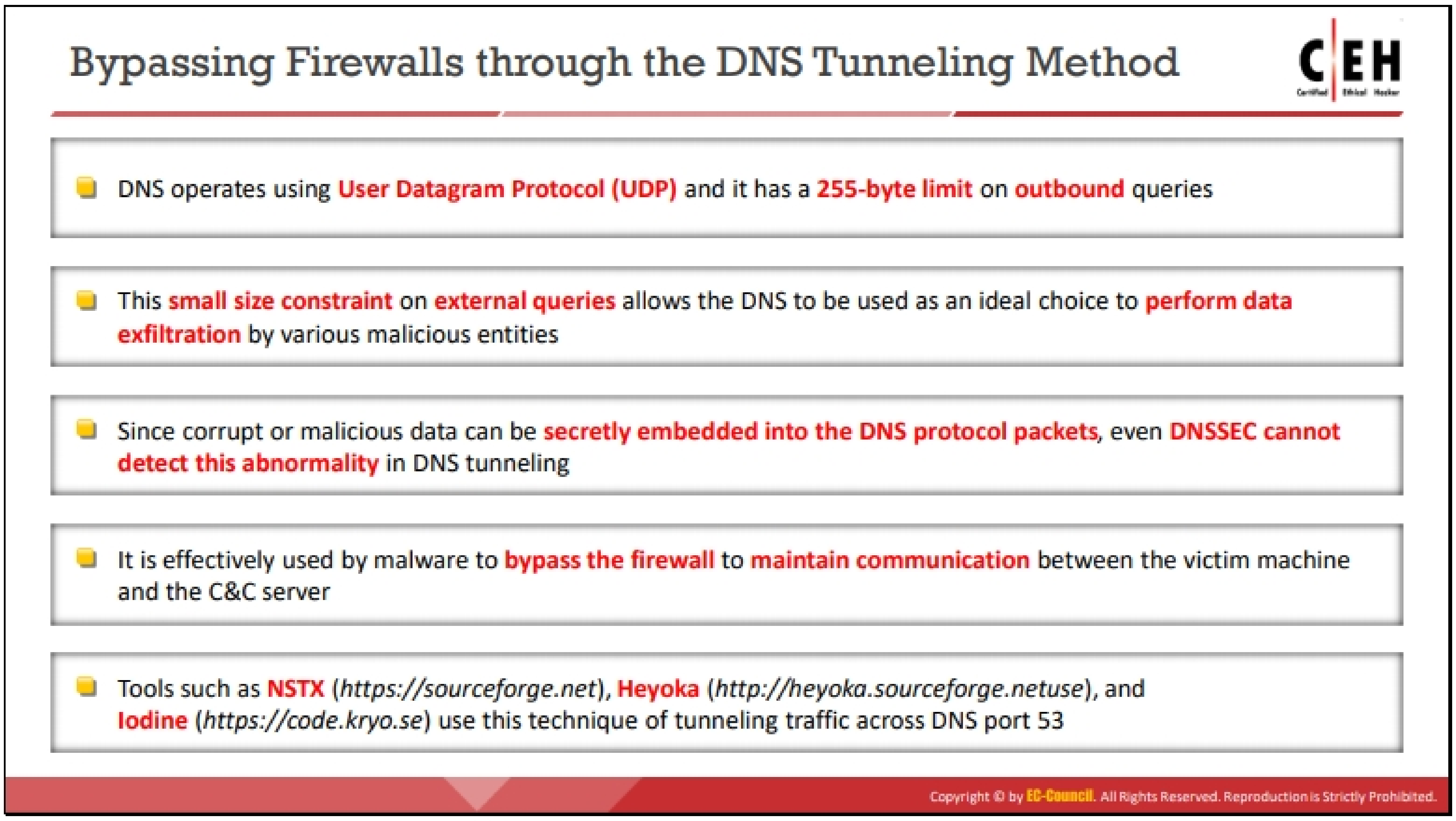
  Secure Pipes is an OS X-based SSH tunneling software. Some of the features of Secure Pipes are as follows:

  - **Remote Forwards**: Selectively open up access to application ports that are usually not easily accessible owing to network or service provider configuration restrictions. Open the door to quickly leverage OS X Server on Internet-facing applications such as email and web hosting.
  - **Local Forwards**: Open application communication ports to remote servers without opening those ports to public networks. Bring the security of VPN communication to clients and servers on an ad hoc basis without the hassle of configuration and management.
  - **SOCKS Proxies**: Easily set up and manage a SOCKS proxy server for either a local client or a whole network to privatize communication and overcome local network restrictions. These tunnels are an indispensable and lightweight tool when traveling abroad, performing digital currency transactions, or simply securing a local network.


Secure Pipes
Connections
General
Info
Connection
Options
Advanced
Cancel
Apply

Connection Name: Digital Ocean Proxy
SSH Server Address: serverc.mydomain.com Port: 22
SSH Username: root Password: •••••••
Local Bind Address: localhost Port: 8080
☐ Automatically configure SOCKS proxy in Network Preferences

Figure 12.52: Screenshot of Secure Pipes



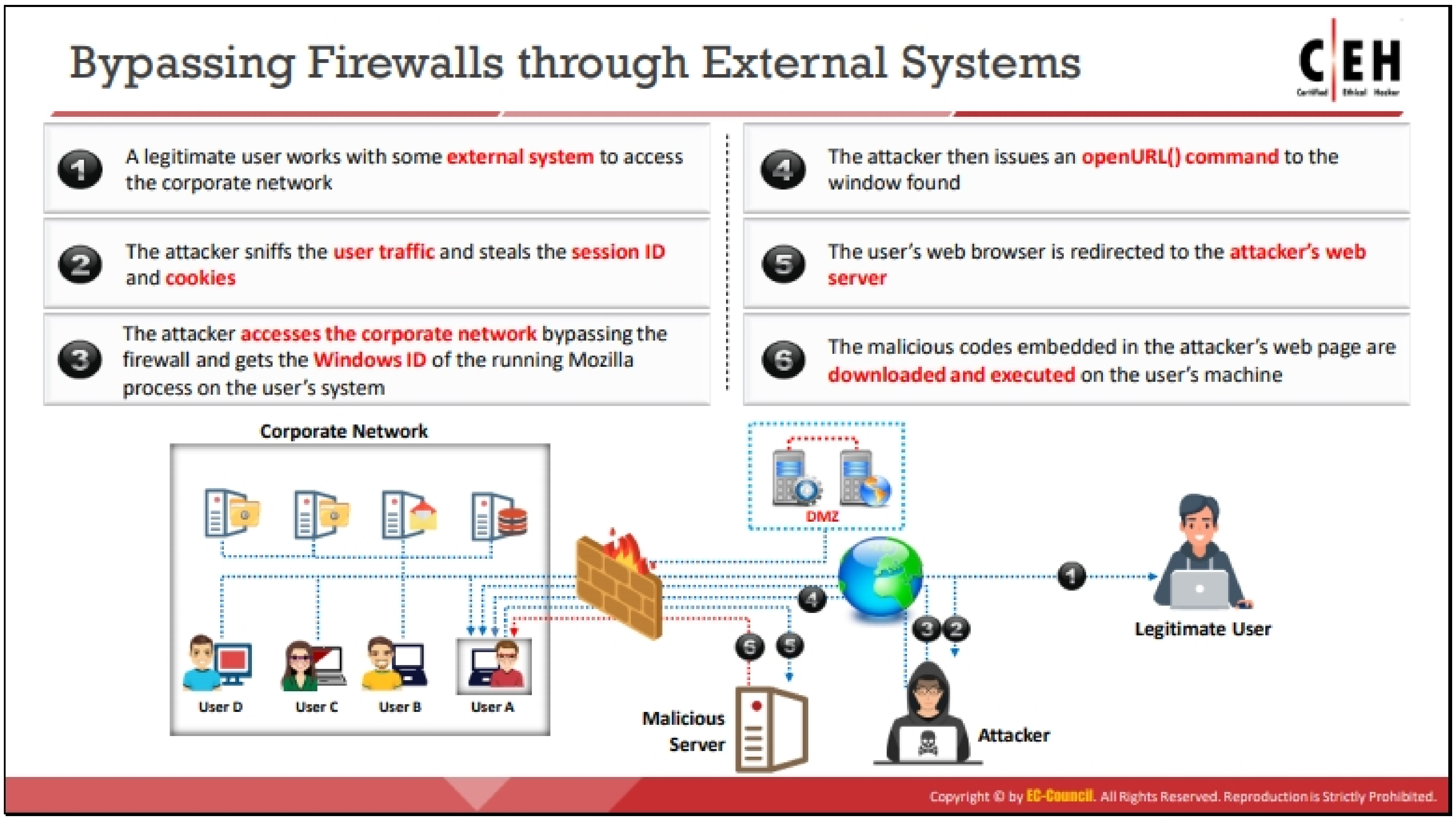
Bypassing Firewalls through the DNS Tunneling Method

- DNS operates using **User Datagram Protocol (UDP)** and it has a **255-byte limit** on **outbound** queries
- This **small size constraint** on **external queries** allows the DNS to be used as an ideal choice to **perform data exfiltration** by various malicious entities
- Since corrupt or malicious data can be **secretly embedded into the DNS protocol packets**, even **DNSSEC cannot detect this abnormality** in DNS tunneling
- It is effectively used by malware to **bypass the firewall** to **maintain communication** between the victim machine and the C&C server
- Tools such as **NSTX** (*https://sourceforge.net*), **Heyoka** (*http://heyoka.sourceforge.netuse*), and **Iodine** (*https://code.kryo.se*) use this technique of tunneling traffic across DNS port 53

Bypassing Firewalls through the DNS Tunneling Method

DNS operates using UDP, and it has a 255-byte limit on outbound queries. Moreover, it allows only alphanumeric characters and hyphens. Such small size constraints on external queries allow DNS to be used as an ideal choice to perform data exfiltration by various malicious entities. Since corrupt or malicious data can be secretly embedded into the DNS protocol packets, even DNSSEC cannot detect the abnormality in DNS tunneling. It is effectively used by malware to bypass the firewall to maintain communication between the victim machine and the C&C server.

Tools such as NSTX (*https://sourceforge.net*), Heyoka (*http://heyoka.sourceforge.netuse*), and Iodine (*https://code.kryo.se*) use this technique of tunneling traffic across DNS port 53.



Bypassing Firewalls through External Systems

1. A legitimate user works with some **external system** to access the corporate network
2. The attacker sniffs the **user traffic** and steals the **session ID** and **cookies**
3. The attacker **accesses the corporate network** bypassing the firewall and gets the **Windows ID** of the running Mozilla process on the user's system
4. The attacker then issues an **openURL() command** to the window found
5. The user's web browser is redirected to the **attacker's web server**
6. The malicious codes embedded in the attacker's web page are **downloaded and executed** on the user's machine

Bypassing Firewalls through External Systems

Attackers can bypass firewall restrictions of target networks from an external system that can access the internal network. This external system can be:

- A home machine of and employee
- A machine that conducts remote administration of the target network
- A machine from the company's network but located at a different place

Steps to be followed to bypass a firewall through external systems:

1. Legitimate user works with some external system to access the corporate network
2. Attacker sniffs the user traffic and steals the session ID and cookies
3. Attacker accesses the corporate network by bypassing the firewall and gets the Windows ID of the running Mozilla process on the user's system
4. Attacker then issues an OpenURL() command to the found window
5. User's web browser is redirected to the attacker's web server
6. The malicious code embedded in the attacker's web page is downloaded and executed on the user's machine

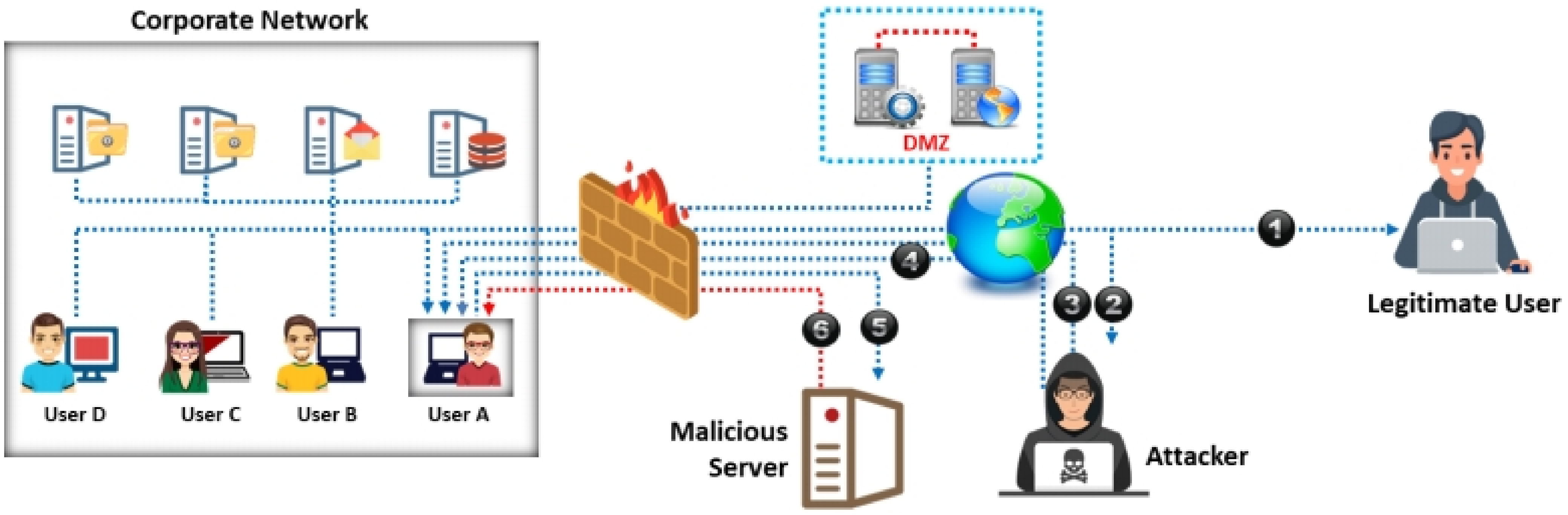
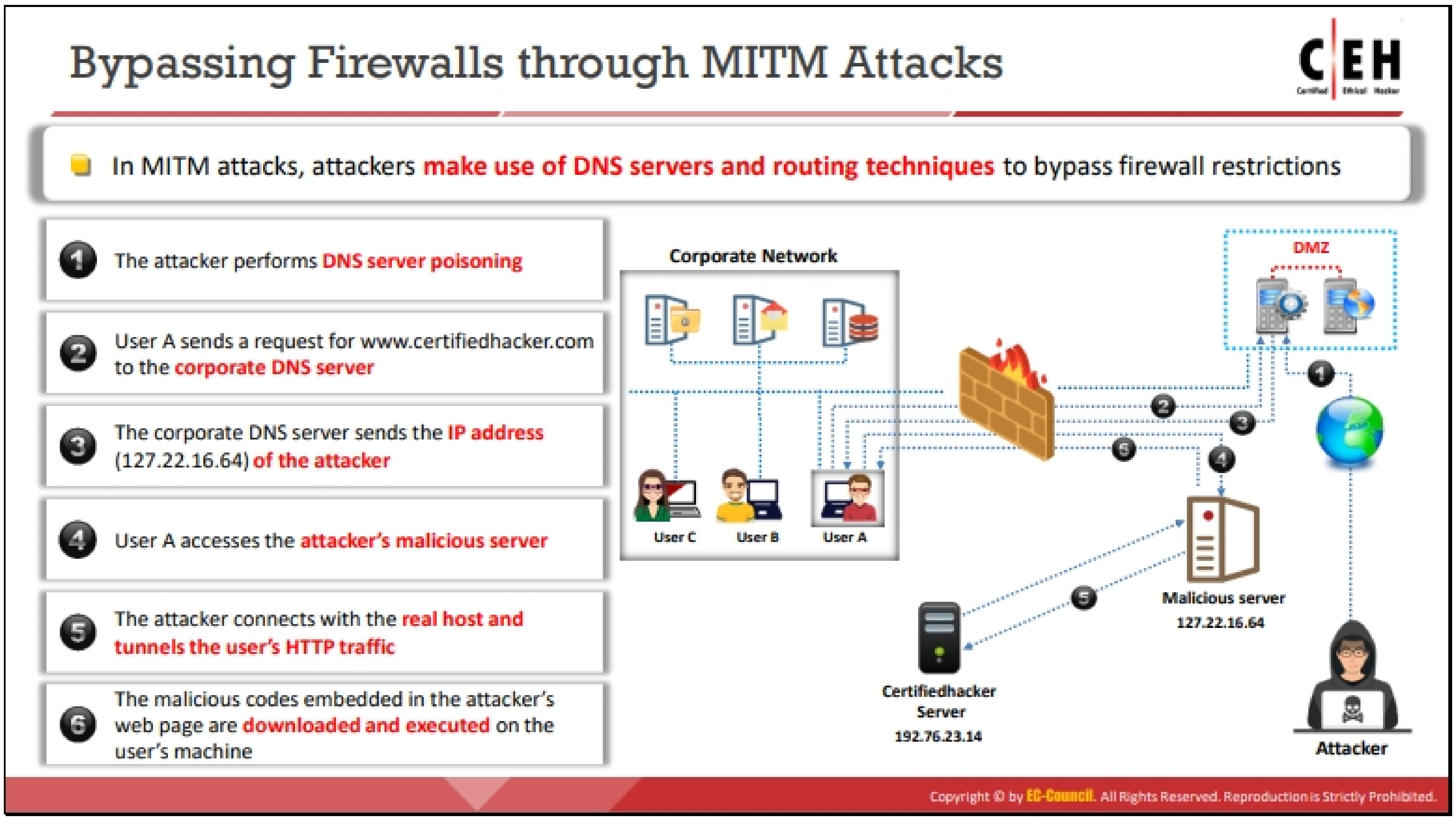

Corporate Network
DMZ
User D
User C
User B
User A
1
2
3
4
5
6
Legitimate User
Malicious Server
Attacker

Figure 12.53: Bypassing firewall through external systems



Bypassing Firewalls through MITM Attacks

- In MITM attacks, attackers **make use of DNS servers and routing techniques** to bypass firewall restrictions

1. The attacker performs **DNS server poisoning**
2. User A sends a request for www.certifiedhacker.com to the **corporate DNS server**
3. The corporate DNS server sends the **IP address** (127.22.16.64) **of the attacker**
4. User A accesses the **attacker's malicious server**
5. The attacker connects with the **real host and tunnels the user's HTTP traffic**
6. The malicious codes embedded in the attacker's web page are **downloaded and executed** on the user's machine

Bypassing Firewalls through MITM Attacks

Most security administrators focus on the possibility of an external or internal network bypassing their firewall while ignoring the fact that firewalls can be bypassed using MITM attacks on DNS servers. In MITM attacks, attackers use DNS servers and routing techniques to bypass firewall restrictions. They may either take over the corporate DNS server or spoof DNS responses to perform the MITM firewall attack.

Steps to be followed to bypass a firewall through MITM attacks:

1. Attacker performs DNS server poisoning
2. User A requests for www.certifiedhacker.com from the corporate DNS server
3. Corporate DNS server sends the IP address (127.22.16.64) of the attacker
4. User A accesses the attacker's malicious server
5. Attacker connects to the real host and tunnels the user's HTTP traffic
6. The malicious code embedded in the attacker's web page is downloaded and executed on the user's machine

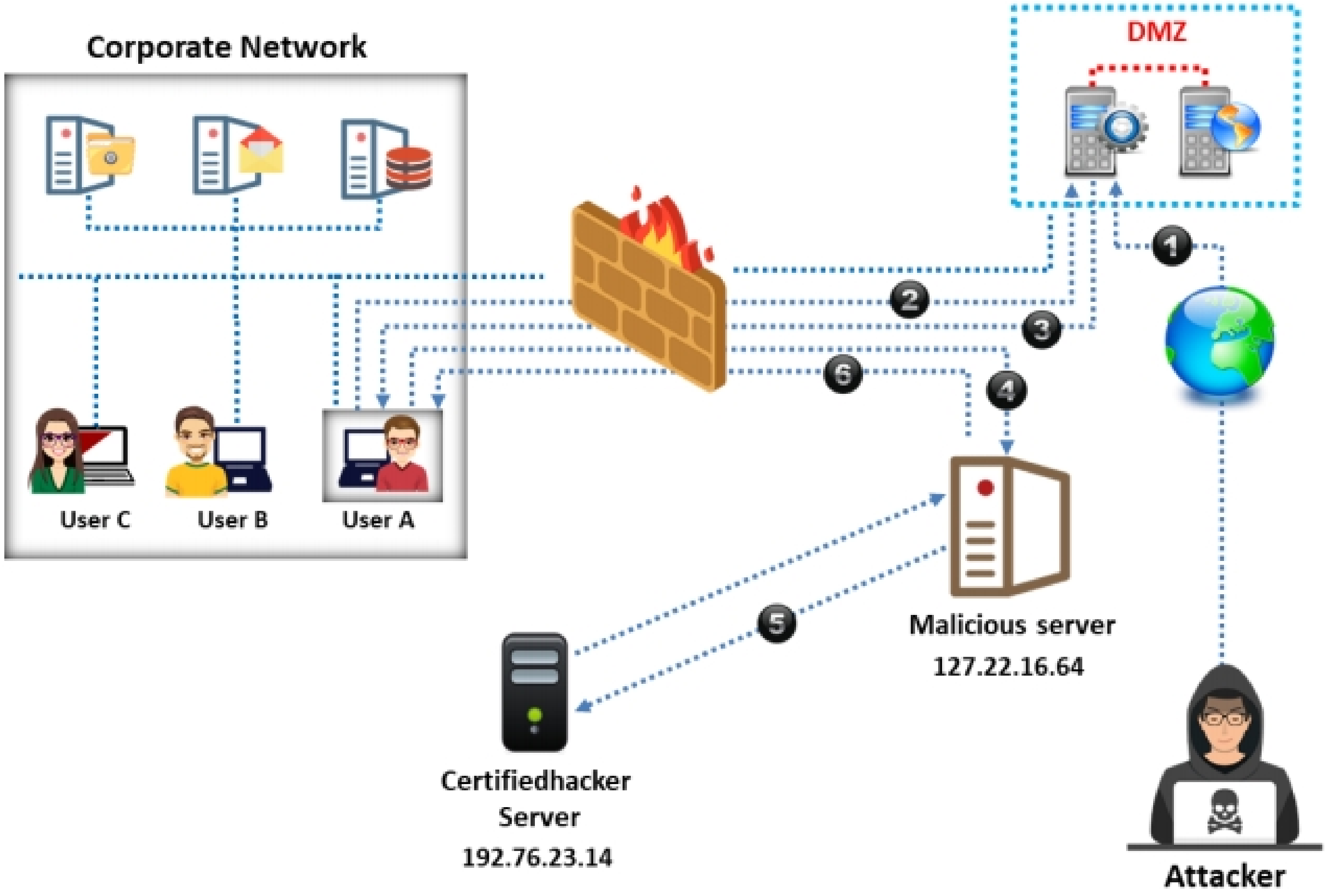
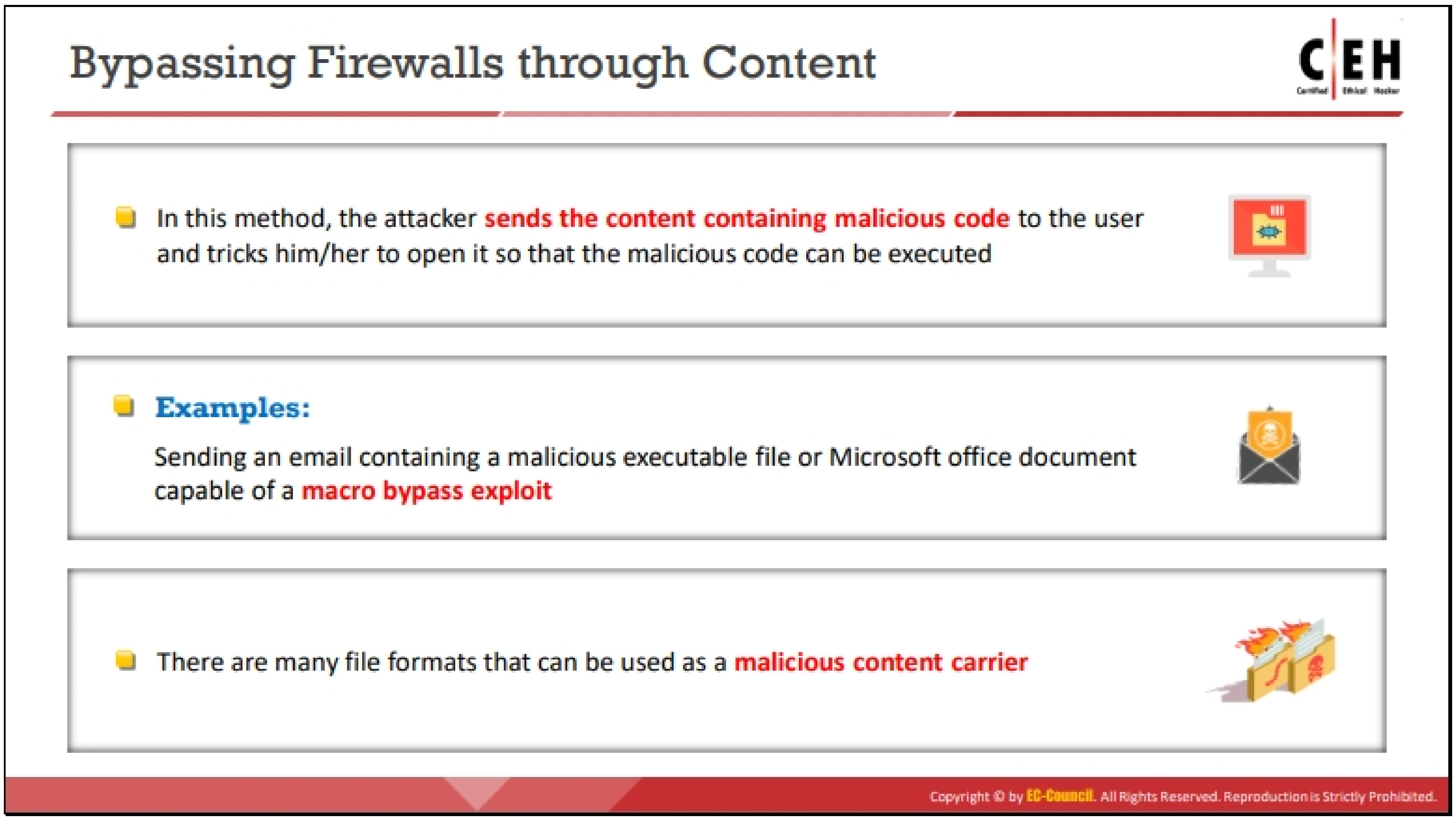

Corporate Network
DMZ
User C
User B
User A
1
2
3
4
5
6
Malicious server
127.22.16.64
Certifiedhacker
Server
192.76.23.14
Attacker

Figure 12.54: Bypassing firewall through MITM attack



Bypassing Firewalls through Content

- In this method, the attacker **sends the content containing malicious code** to the user and tricks him/her to open it so that the malicious code can be executed
- **Examples:**

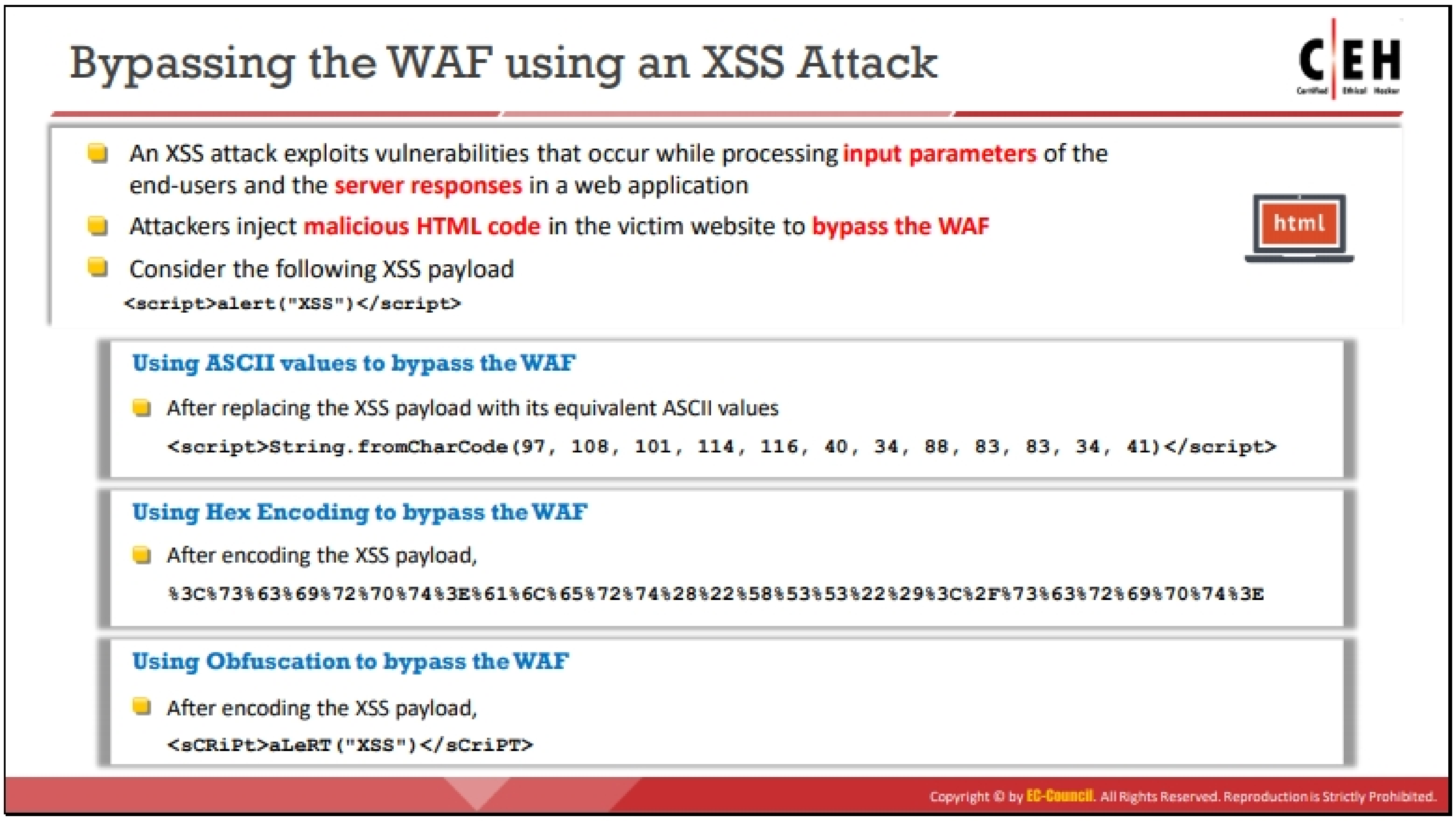
  Sending an email containing a malicious executable file or Microsoft office document capable of a **macro bypass exploit**
- There are many file formats that can be used as a **malicious content carrier**

Bypassing Firewalls through Content

In this method, the attacker sends content containing malicious code to the user and tricks him/her into opening it so that the malicious code can be executed. For example, an attacker can send an email containing a malicious executable file or Microsoft office document capable of exploiting a macro bypass exploit. Attackers can also target WWW/FTP servers and embed Trojan horse files as software installation files, mobile phone software, and so on to lure users into accessing them. There are many file formats for text, multimedia, and graphics content that can be used to carry malicious content.

Commonly used file formats for carrying malicious content are:

- EXE, COM, BAT, PS, PDF CDR (Corel Draw)
- DVB, DWG (AutoCAD)
- SMM (AMI Pro)
- DOC, DOT, CNV, ASD (MS Word)
- XLS, XLB, XLT (MS Excel)
- ADP, MDA, MDB, MDE, MDN, MDZ (MS Access)
- VSD (Visio)
- MPP, MPT (MS Project)
- PPT, PPS, POT (MS PowerPoint)
- MSG, OTM (MS Outlook)



Bypassing the WAF using an XSS Attack

- An XSS attack exploits vulnerabilities that occur while processing **input parameters** of the end-users and the **server responses** in a web application
- Attackers inject **malicious HTML code** in the victim website to **bypass the WAF**
- Consider the following XSS payload

`<script>alert("XSS")</script>`

Using ASCII values to bypass the WAF

- After replacing the XSS payload with its equivalent ASCII values

`<script>String.fromCharCode(97, 108, 101, 114, 116, 40, 34, 88, 83, 83, 34, 41)</script>`

Using Hex Encoding to bypass the WAF

- After encoding the XSS payload,

`%3C%73%63%69%72%70%74%3E%61%6C%65%72%74%28%22%58%53%53%22%29%3C%2F%73%63%72%69%70%74%3E`

Using Obfuscation to bypass the WAF

- After encoding the XSS payload,

`<sCRiPt>aLeRT("XSS")</sCriPT>`

Bypassing the WAF using an XSS Attack

XSS attack exploits vulnerabilities that occur while processing the input parameters of end users and the server responses in a web application. Attackers take advantage of these vulnerabilities to inject malicious HTML code into the victim website to bypass the WAF.

- **Using ASCII values to bypass the WAF**

  In this technique, attackers use ASCII characters to bypass the WAF. For example, consider the following XSS payload

  ```
  <script>alert("XSS")</script>
  ```

  When the above JavaScript code is executed, the WAF filters escape single quotes, double magic quotes, etc. Hence, the above payload is filtered by the WAF. To bypass the WAF, we need to convert the above payload into its equivalent ASCII values and then execute it. The JavaScript will automatically convert the ASCII values back into the original characters. Attackers use online websites to convert an XSS payload into its ASCII equivalent. Alternatively, the Hackbar Mozilla addon can be used to get ASCII values.

  Consider the XSS payload given below:

  ```
  XSS Payload:alert("XSS")
  ```

  The equivalent ASCII values are:

  ```
  String.fromCharCode(97, 108, 101, 114, 116, 40, 34, 88, 83, 83, 34, 41)
  ```

  The above values are inserted into the XSS payload:

  ```
  <script>String.fromCharCode(97, 108, 101, 114, 116, 40, 34, 88, 83, 83,
  34, 41)</script>
  ```

  The above payload bypasses the WAF filters successfully.

- **Using Hex Encoding to bypass the WAF**

  In this technique, the entire XSS payload is replaced with Hex values to bypass WAF. Attackers use online websites such as *http://www.convertstring.com/EncodeDecode/HexEncode* to convert the XSS payload into equivalent Hex values. For example, consider the following XSS payload

  ```
  <script>alert("XSS")</script>
  ```

  The encoded value for the XSS payload is

  ```
  %3C%73%63%69%72%70%74%3E%61%6C%65%72%74%28%22%58%53%53%22%29%3C%2F%73%63%72%69%70%74%3E
  ```

  The above payload bypasses the WAF filters successfully.

- **Using Obfuscation to bypass the WAF**

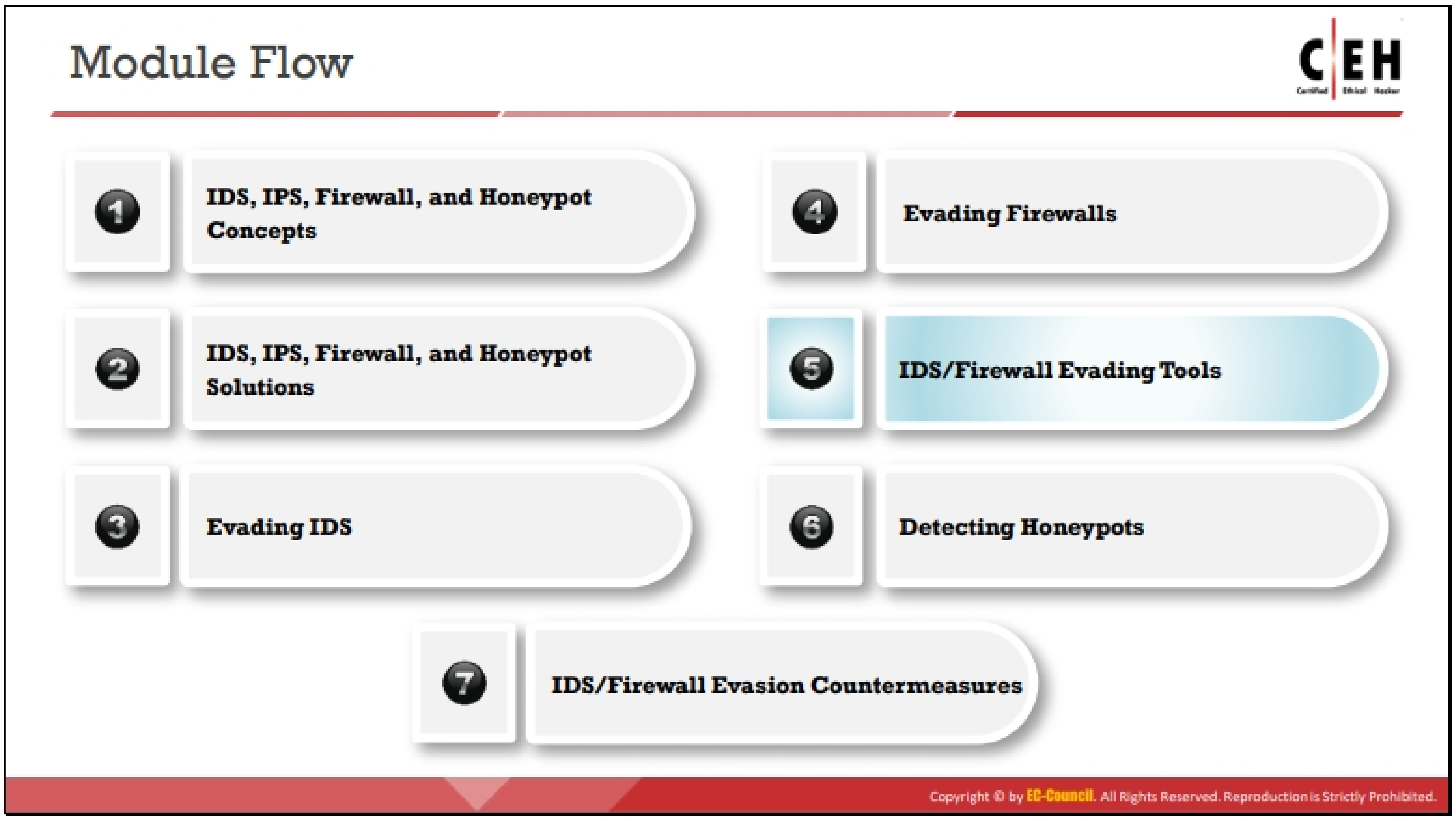
  Attackers use the obfuscation technique to bypass the WAF. In this technique, attackers use a combination of upper- and lower-case letters in the XSS payload. For example, consider the following XSS payload:

  ```
  <script>alert("XSS")</script>
  ```

  Using obfuscation, the above payload is replaced with

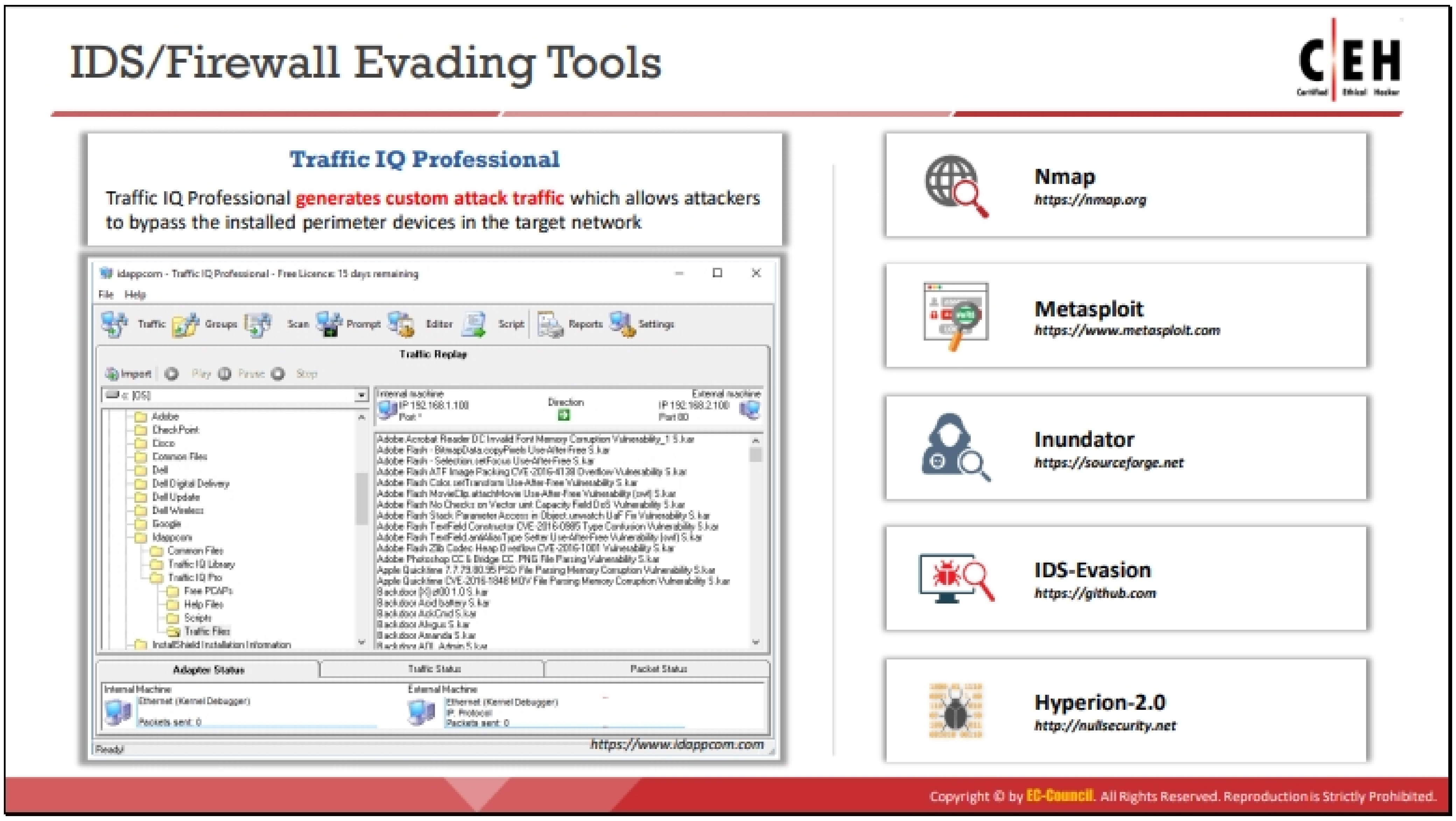
  ```
  <sCRiPt>aLeRT("XSS")</sCriPT>
  ```

  The above payload bypasses the WAF successfully



Module Flow

1. IDS, IPS, Firewall, and Honeypot Concepts
2. IDS, IPS, Firewall, and Honeypot Solutions
3. Evading IDS
4. Evading Firewalls
5. **IDS/Firewall Evading Tools**
6. Detecting Honeypots
7. IDS/Firewall Evasion Countermeasures



IDS/Firewall Evading Tools

Traffic IQ Professional

Traffic IQ Professional **generates custom attack traffic** which allows attackers to bypass the installed perimeter devices in the target network

idappcom - Traffic IQ Professional - Free Licence: 15 days remaining
File
Help
Traffic
Groups
Scan
Prompt
Editor
Script
Reports
Settings
Traffic Replay
Import
Play
Pause
Stop
Adapter Status
Traffic Status
Packet Status
Ready

https://www.idappcom.com

- **Nmap** https://nmap.org
- **Metasploit** https://www.metasploit.com
- **Inundator** https://sourceforge.net
- **IDS-Evasion** https://github.com
- **Hyperion-2.0** http://nullsecurity.net

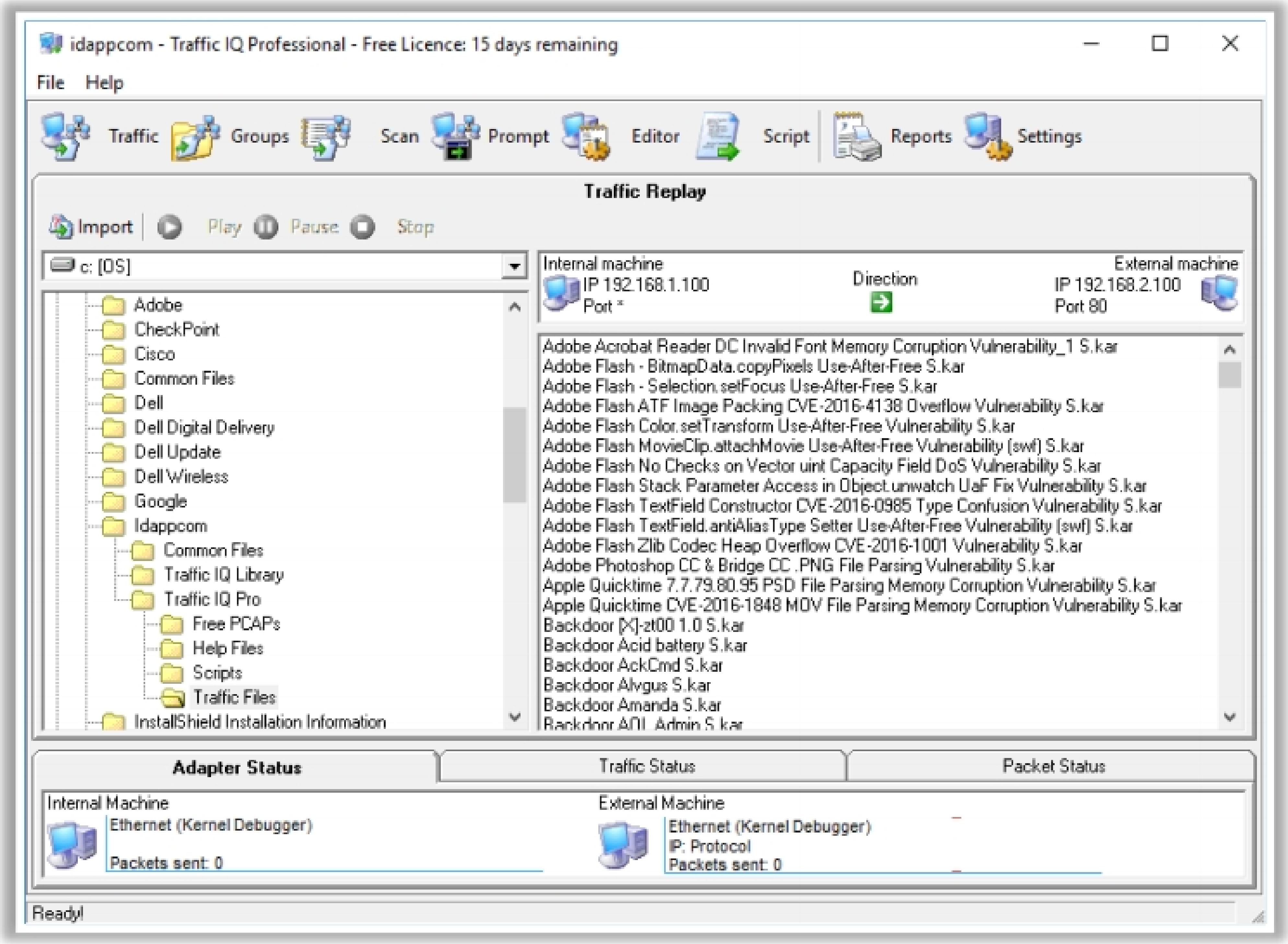
IDS/Firewall Evading Tools

During firewall evasion, attackers use various security-auditing tools that assess firewall behavior. This section lists some of these tools that help attackers to bypass firewall restrictions. They automate the process of bypassing firewall rules while increasing effectiveness and consuming less time.

- **Traffic IQ Professional**

  Source: *https://www.idappcom.com*

  Traffic IQ Professional is a tool that audits and validates the behavior of security devices by generating the standard application traffic or attack traffic between two virtual machines. This tool is generally used by security personnel for assessing, auditing, and testing the behavioral characteristics of any non-proxy packet-filtering device, which can include application firewalls, IDS, IPS, routers, switches, etc. However, as this tool can generate custom attack traffic, it is extensively employed by attackers to bypass the installed perimeter devices in the target network.


idappcom - Traffic IQ Professional - Free Licence: 15 days remaining
File
Help
Traffic
Groups
Scan
Prompt
Editor
Script
Reports
Settings
Traffic Replay
Import
Play
Pause
Stop
c: [OS]
Adobe
CheckPoint
Cisco
Common Files
Dell
Dell Digital Delivery
Dell Update
Dell Wireless
Google
Idappcom
Common Files
Traffic IQ Library
Traffic IQ Pro
Free PCAPs
Help Files
Scripts
Traffic Files
InstallShield Installation Information
Adapter Status
Traffic Status
Packet Status
Ready!

Internal machine
IP 192.168.1.100
Port *

Direction

External machine
IP 192.168.2.100
Port 80

Adobe Acrobat Reader DC Invalid Font Memory Corruption Vulnerability_1 S.kar
Adobe Flash - BitmapData.copyPixels Use-After-Free S.kar
Adobe Flash - Selection.setFocus Use-After-Free S.kar
Adobe Flash ATF Image Packing CVE-2016-4138 Overflow Vulnerability S.kar
Adobe Flash Color.setTransform Use-After-Free Vulnerability S.kar
Adobe Flash MovieClip.attachMovie Use-After-Free Vulnerability (swf) S.kar
Adobe Flash No Checks on Vector uint Capacity Field DoS Vulnerability S.kar
Adobe Flash Stack Parameter Access in Object.unwatch UaF Fix Vulnerability S.kar
Adobe Flash TextField Constructor CVE-2016-0985 Type Confusion Vulnerability S.kar
Adobe Flash TextField.antiAliasType Setter Use-After-Free Vulnerability (swf) S.kar
Adobe Flash Zlib Codec Heap Overflow CVE-2016-1001 Vulnerability S.kar
Adobe Photoshop CC & Bridge CC .PNG File Parsing Vulnerability S.kar
Apple Quicktime 7.7.79.80.95 PSD File Parsing Memory Corruption Vulnerability S.kar
Apple Quicktime CVE-2016-1848 MOV File Parsing Memory Corruption Vulnerability S.kar
Backdoor [X]-zt00 1.0 S.kar
Backdoor Acid battery S.kar
Backdoor AckCmd S.kar
Backdoor Alvgus S.kar
Backdoor Amanda S.kar
Backdoor AOL Admin S.kar

Internal Machine
Ethernet (Kernel Debugger)
Packets sent: 0

External Machine
Ethernet (Kernel Debugger)
IP: Protocol
Packets sent: 0

Figure 12.55: Screenshot of Traffic IQ Professional

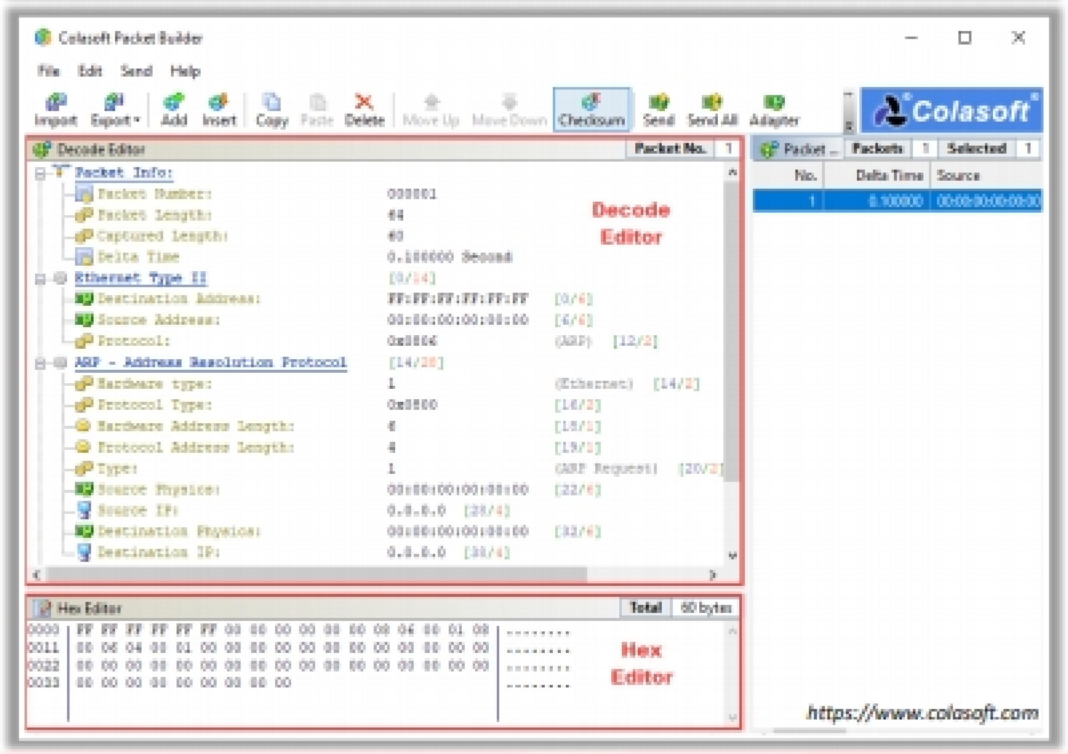
Some additional IDS/firewall evasion tools are as follows:

- Nmap (*https://nmap.org*)
- Metasploit (*https://www.metasploit.com*)
- Inundator (*https://sourceforge.net*)
- IDS-Evasion (*https://github.com)*
- Hyperion-2.0 (*http://nullsecurity.net)*

Packet Fragment Generator Tools

Colasoft Packet Builder

Colasoft Packet Builder is used to create custom network packets and fragmenting packets. Attackers use this tool to **create custom malicious packets** and fragment them in such a way that firewalls will not detect them


Colasoft Packet Builder
File Edit Send Help
Import Export Add Insert Copy Paste Delete Move Up Move Down Checksum Send Send All Adapter
Colasoft
Decode Editor
Packet No. 1
Decode Editor
Hex Editor
Total 60 bytes
Hex Editor
Packet ... Packets 1 Selected 1
No. Delta Time Source

```
Packet Info:
  Packet Number:                000001
  Packet Length:                64
  Captured Length:              60
  Delta Time                    0.100000 Second
Ethernet Type II                [0/14]
  Destination Address:          FF:FF:FF:FF:FF:FF   [0/6]
  Source Address:               00:00:00:00:00:00   [6/6]
  Protocol:                     0x0806              (ARP)  [12/2]
ARP - Address Resolution Protocol  [14/28]
  Hardware type:                1                   (Ethernet)  [14/2]
  Protocol Type:                0x0800              [16/2]
  Hardware Address Length:      6                   [18/1]
  Protocol Address Length:      4                   [19/1]
  Type:                         1                   (ARP Request)  [20/2]
  Source Physics:               00:00:00:00:00:00   [22/6]
  Source IP:                    0.0.0.0  [28/4]
  Destination Physics:          00:00:00:00:00:00   [32/6]
  Destination IP:               0.0.0.0  [38/4]
```

```
0000 | FF FF FF FF FF FF 00 00 00 00 00 00 08 06 00 01 08 | ........
0011 | 00 06 04 00 01 00 00 00 00 00 00 00 00 00 00 00 00 | ........
0022 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ........
0033 | 00 00 00 00 00 00 00 00 00                         | ........
```

```
1    0.100000    00:00:00:00:00:00
```

https://www.colasoft.com

CommView
https://www.tamos.com

NetScanTools Pro
https://www.netscantools.com

Ostinato
https://ostinato.org

WAN Killer
https://www.solarwinds.com

WireEdit
https://wireedit.com

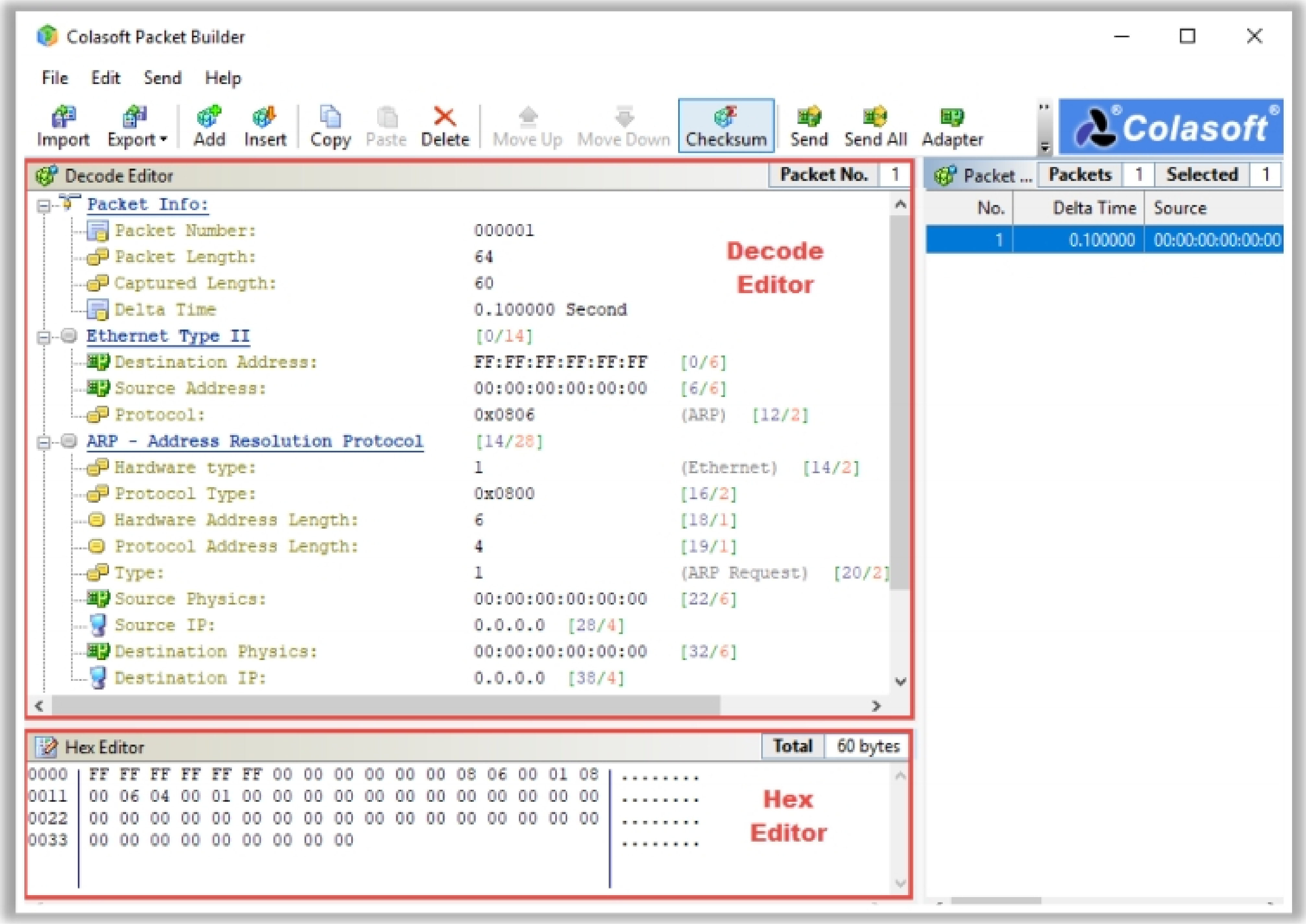
Packet Fragment Generator Tools

There are various packet fragment generators that attackers use to perform fragmentation attacks on firewalls to bypass them.

- **Colasoft Packet Builder**

  Source: *https://www.colasoft.com*

  Colasoft Packet Builder is used to create custom network packets and fragmenting packets. Attackers use this tool to create custom malicious packets and fragment them such that firewalls cannot detect them. They can create custom network packets such as Ethernet Packet, ARP Packet, IP Packet, TCP Packet, and UDP Packet. Security professionals use this tool to check your network's protection against attacks and intruders.


Colasoft Packet Builder
File
Edit
Send
Help
Import
Export
Add
Insert
Copy
Paste
Delete
Move Up
Move Down
Checksum
Send
Send All
Adapter
Colasoft
Decode Editor
Packet No. 1
Packet ...
Packets 1
Selected 1
No.
Delta Time
Source
1
0.100000
00:00:00:00:00:00
Decode Editor
Hex Editor
Total 60 bytes
Hex Editor

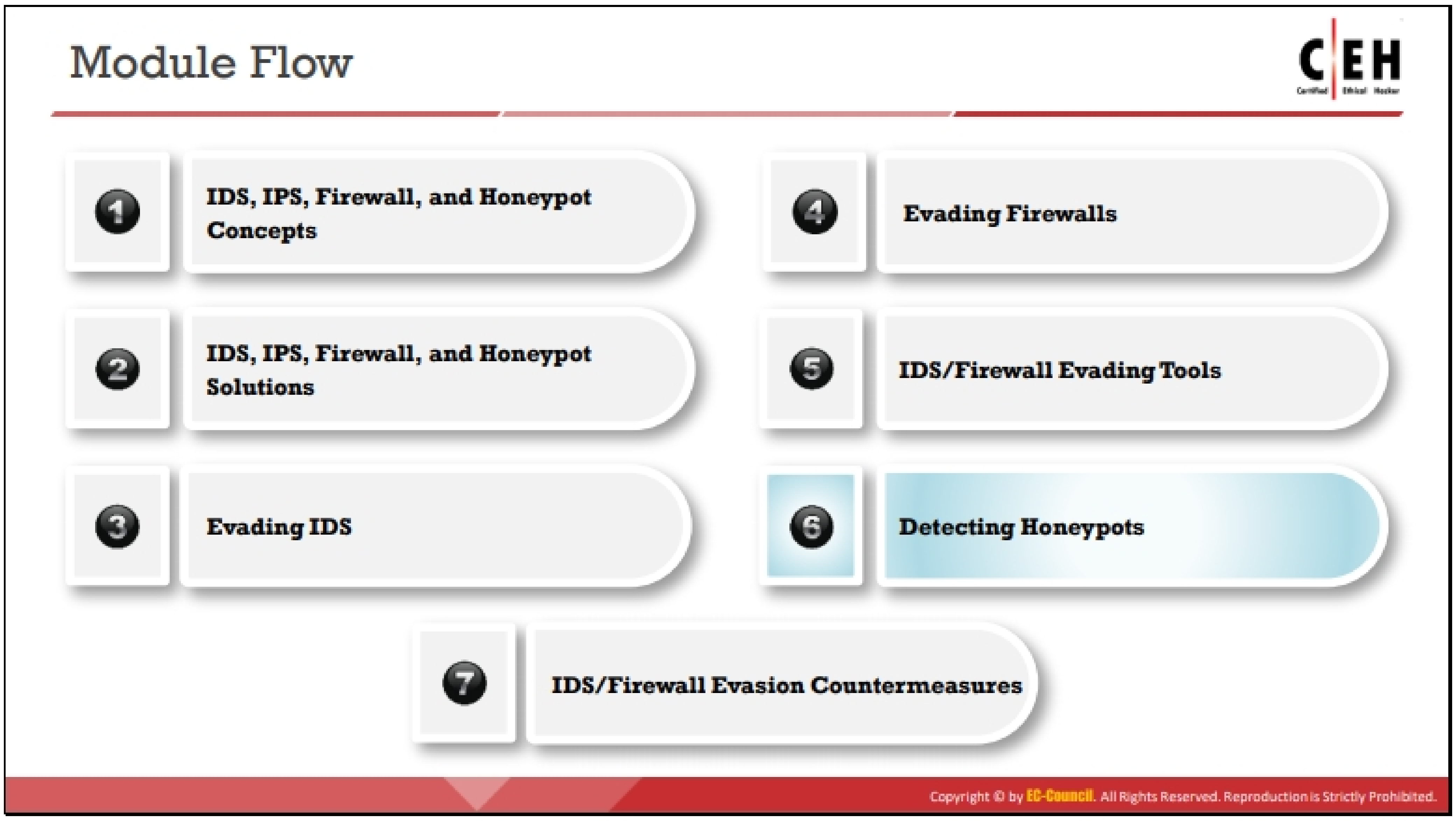
```
Packet Info:
    Packet Number:              000001
    Packet Length:              64
    Captured Length:            60
    Delta Time                  0.100000 Second
Ethernet Type II                [0/14]
    Destination Address:        FF:FF:FF:FF:FF:FF   [0/6]
    Source Address:             00:00:00:00:00:00   [6/6]
    Protocol:                   0x0806              (ARP)   [12/2]
ARP - Address Resolution Protocol   [14/28]
    Hardware type:              1                   (Ethernet)   [14/2]
    Protocol Type:              0x0800              [16/2]
    Hardware Address Length:    6                   [18/1]
    Protocol Address Length:    4                   [19/1]
    Type:                       1                   (ARP Request)   [20/2]
    Source Physics:             00:00:00:00:00:00   [22/6]
    Source IP:                  0.0.0.0   [28/4]
    Destination Physics:        00:00:00:00:00:00   [32/6]
    Destination IP:             0.0.0.0   [38/4]
```

```
0000 | FF FF FF FF FF FF 00 00 00 00 00 00 08 06 00 01 08 | ........
0011 | 00 06 04 00 01 00 00 00 00 00 00 00 00 00 00 00 00 | ........
0022 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ........
0033 | 00 00 00 00 00 00 00 00 00                         | ........
```

Figure 12.56: Screenshot of Colasoft Packet Builder

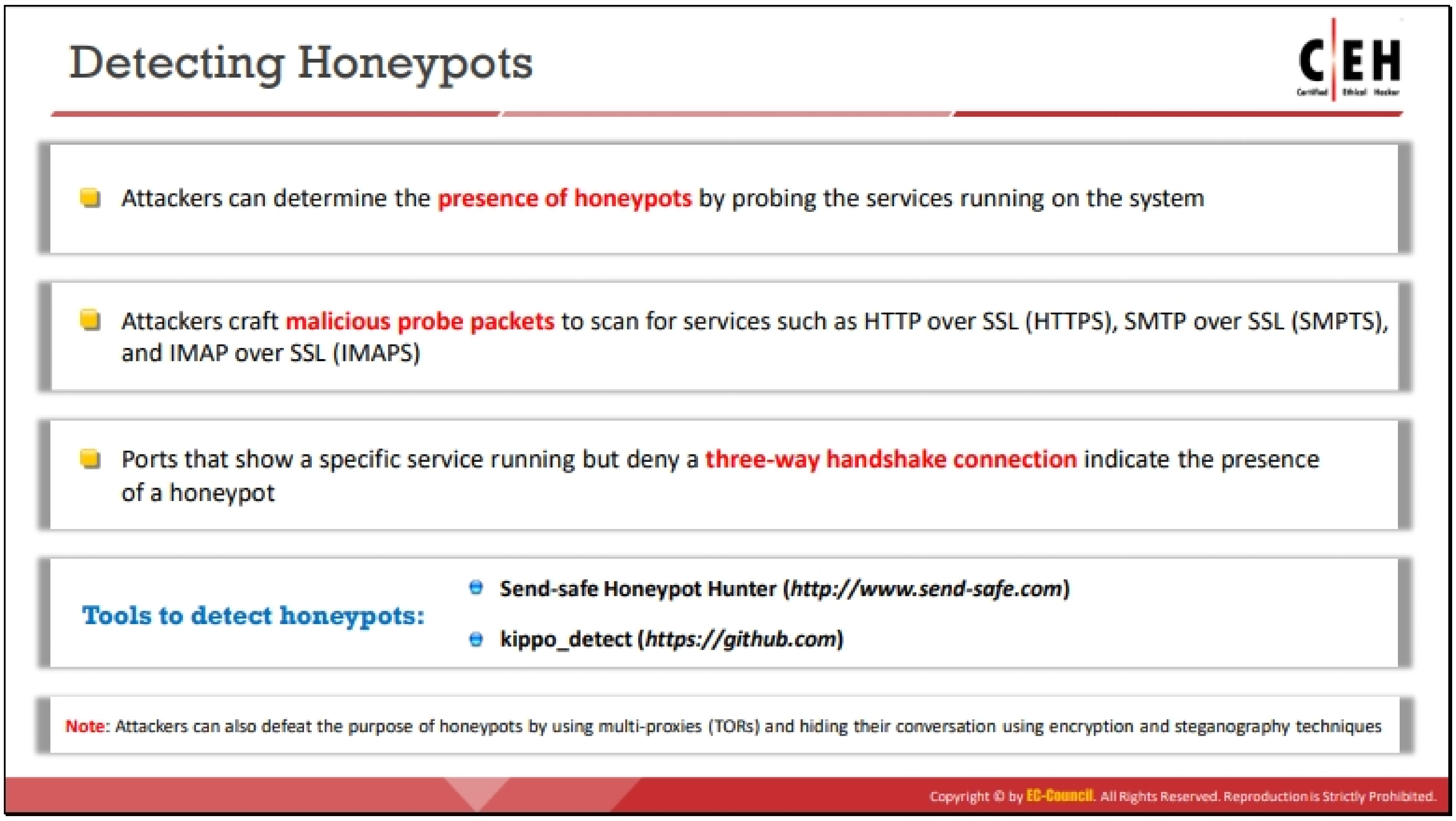
Some additional packet generator tools are listed below:

- CommView (*https://www.tamos.com*)
- NetScanTools Pro (*https://www.netscantools.com*)
- Ostinato (*https://ostinato.org*)
- WAN Killer (*https://www.solarwinds.com*)
- WireEdit (*https://wireedit.com*)



Module Flow

1. IDS, IPS, Firewall, and Honeypot Concepts
2. IDS, IPS, Firewall, and Honeypot Solutions
3. Evading IDS
4. Evading Firewalls
5. IDS/Firewall Evading Tools
6. Detecting Honeypots
7. IDS/Firewall Evasion Countermeasures



Detecting Honeypots

- Attackers can determine the **presence of honeypots** by probing the services running on the system
- Attackers craft **malicious probe packets** to scan for services such as HTTP over SSL (HTTPS), SMTP over SSL (SMPTS), and IMAP over SSL (IMAPS)
- Ports that show a specific service running but deny a **three-way handshake connection** indicate the presence of a honeypot

Tools to detect honeypots:

- **Send-safe Honeypot Hunter (*http://www.send-safe.com*)**
- **kippo_detect (*https://github.com*)**

Note: Attackers can also defeat the purpose of honeypots by using multi-proxies (TORs) and hiding their conversation using encryption and steganography techniques

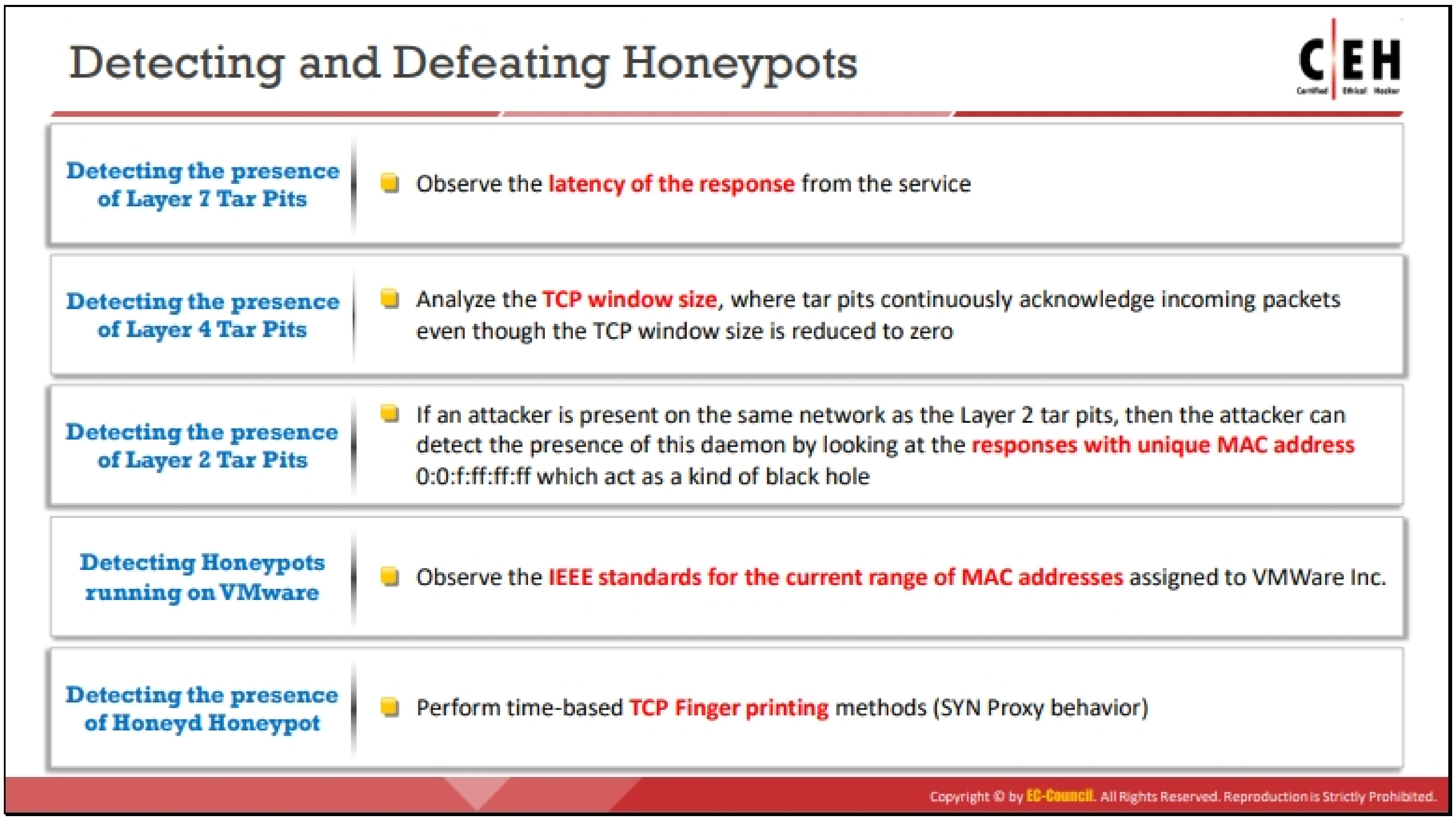
Detecting Honeypots

Honeypots are traps set to detect, deflect, or counteract unauthorized intrusion attempts. While attempting to break into the target network, attackers perform honeypot detection using various tools and techniques. This section discusses these tools and how they are used.

A honeypot is an Internet system designed primarily for diverting attackers by tricking or attracting them during their attempts to gain unauthorized access to information systems.

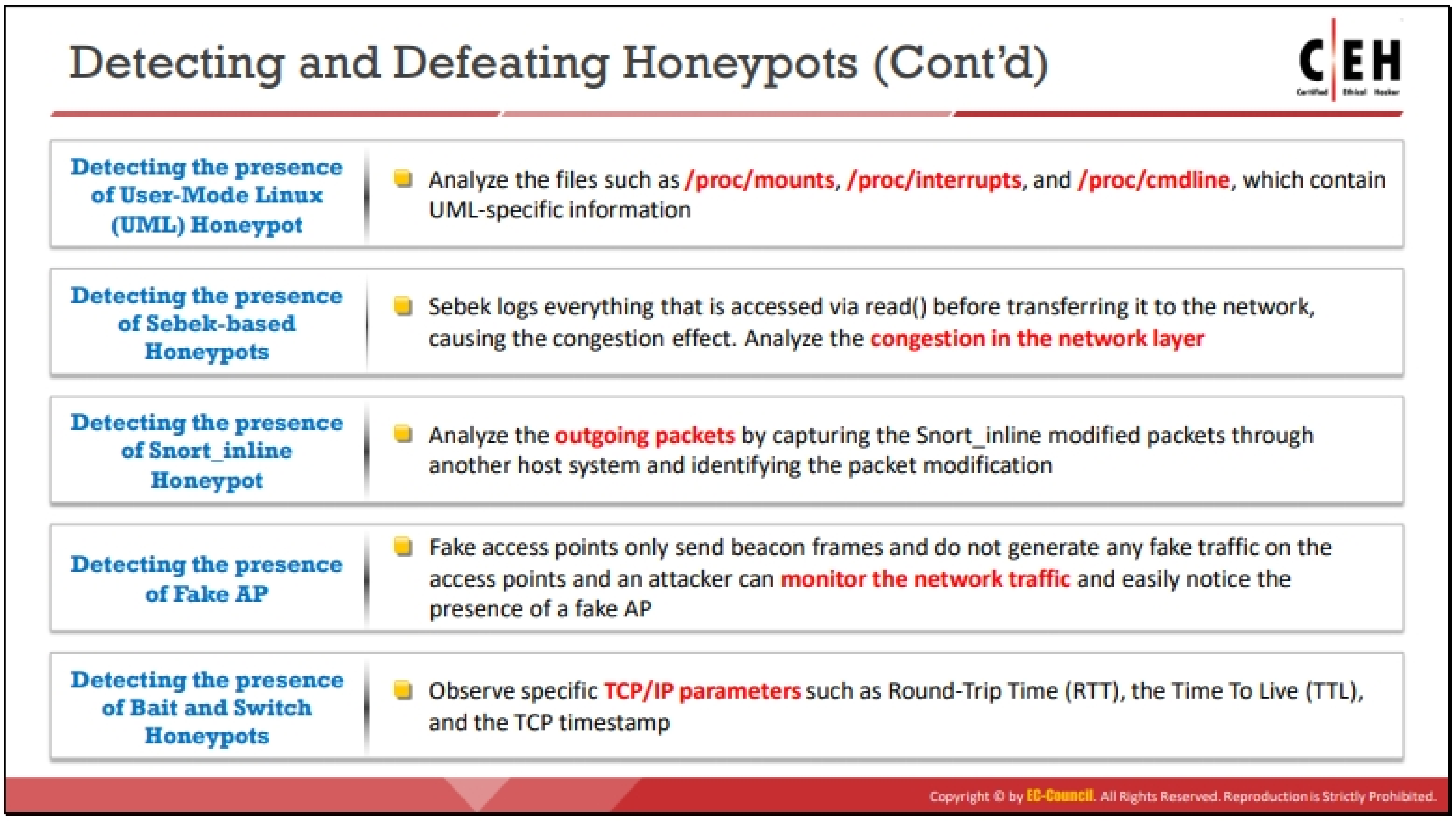
Attackers can determine the presence of honeypots by probing the services running on the system. Attackers use honeypot detection systems or methods to identify the honeypots installed on the target network. They craft malicious probe packets to scan for services such as HTTP over SSL (HTTPS), SMTP over SSL (SMPTS), and IMAP over SSL (IMAPS). Ports that show a particular service running but deny a three-way handshake connection indicate the presence of a honeypot. Once they detect honeypots, attackers try to bypass them so that they can focus on targeting the actual network. Tools to detect honeypots include Send-safe Honeypot Hunter (*http://www.send-safe.com*) and kippo_detect (*https://github.com*).

Note: Attackers can also defeat honeypots by using multi-proxies (TORs) and hiding their conversation using encryption and steganography techniques.



Detecting and Defeating Honeypots

Detecting the presence of Layer 7 Tar Pits	- Observe the **latency of the response** from the service
Detecting the presence of Layer 4 Tar Pits	- Analyze the **TCP window size**, where tar pits continuously acknowledge incoming packets even though the TCP window size is reduced to zero
Detecting the presence of Layer 2 Tar Pits	- If an attacker is present on the same network as the Layer 2 tar pits, then the attacker can detect the presence of this daemon by looking at the **responses with unique MAC address** 0:0:f:ff:ff:ff which act as a kind of black hole
Detecting Honeypots running on VMware	- Observe the **IEEE standards for the current range of MAC addresses** assigned to VMWare Inc.
Detecting the presence of Honeyd Honeypot	- Perform time-based **TCP Finger printing** methods (SYN Proxy behavior)



Detecting and Defeating Honeypots (Cont'd)

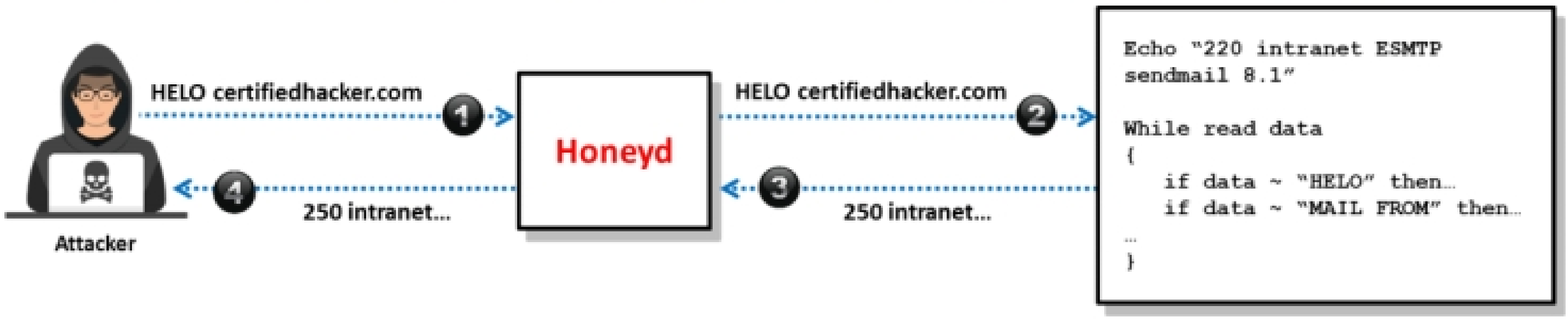
Detecting the presence of User-Mode Linux (UML) Honeypot	- Analyze the files such as **/proc/mounts**, **/proc/interrupts**, and **/proc/cmdline**, which contain UML-specific information
Detecting the presence of Sebek-based Honeypots	- Sebek logs everything that is accessed via read() before transferring it to the network, causing the congestion effect. Analyze the **congestion in the network layer**
Detecting the presence of Snort_inline Honeypot	- Analyze the **outgoing packets** by capturing the Snort_inline modified packets through another host system and identifying the packet modification
Detecting the presence of Fake AP	- Fake access points only send beacon frames and do not generate any fake traffic on the access points and an attacker can **monitor the network traffic** and easily notice the presence of a fake AP
Detecting the presence of Bait and Switch Honeypots	- Observe specific **TCP/IP parameters** such as Round-Trip Time (RTT), the Time To Live (TTL), and the TCP timestamp

Detecting and Defeating Honeypots

A honeypot is a security mechanism that is deployed to counterattack and trap attackers. Honeypots lure attackers into performing malicious activities, and this attack information provides insights into the level and type of threats a network infrastructure can face. As an attacker, determining whether the target system is a legitimate one or a honeypot is essential to compromise the network without being detected. Identifying and defeating these honeypot establishments stealthily is the fundamental task of a professional hacker.

Some techniques used to identify, detect, and defeat various honeypot infrastructures are discussed below:

- **Detecting the presence of Layer 7 Tar Pits**: Tar pits are security entities that are similar to honeypots, which are designed to respond slowly to incoming requests. They slow down unauthorized attempts of hackers. Layer 7 tar pits react slowly to incoming SMTP commands by attackers/spammers. Attackers can identify the presence of Layer 7 tar pits by looking at the latency of the response from the service.
- **Detecting the presence of Layer 4 Tar Pits**: Layer 4 tar pits manipulate the TCP/IP stack and are effectively employed to slow down the spreading of worms, backdoors, etc. In these tar pits, the iptables accept the incoming TCP/IP connection and spontaneously switch to a zero-window size, blocking the attacker from sending further data. This connection cannot be terminated by the attacker, as no data is transferred to the target machine. Layer 4 tar pits such as Labrea can be identified by the attacker by analyzing the TCP window size, where the tar pit continuously acknowledges incoming packets even though the TCP window size is reduced to zero.
- **Detecting the presence of Layer 2 Tar Pits**: If an attacker launches an attack from the same network, the issue of Layer 2 arises. Layer 2 tar pits are used to block the network penetration of the attacker who gains access to the network as well as to prevent internal threats. The attacker can detect the presence of this daemon by looking at the responses with the unique MAC address 0:0:f:ff:ff:ff, which acts as a kind of black hole. An attacker can also identify the presence of these tar pits by analyzing the ARP responses.
- **Detecting Honeypots running on VMware**: VMWare is a commercially available virtual machine that is used to launch multiple instances of an OS simultaneously. These virtual machines can be configured with various virtual machine resources such as CPU, memory, disks, I/O devices, etc. Owing to its numerous advantages, VMWare is widely used to launch honeypots. Attackers can identify instances that are running on the VMWare virtual machine by analyzing the MAC address. By looking at the IEEE standards for the current range of MAC addresses assigned to VMWare Inc., an attacker can identify the presence of VMWare-based honeypots.
- **Detecting the presence of Honeyd Honeypot**: Honeyd is a widely used honeypot daemon. It is used to create thousands of honeypots easily. It is a network-simulated and service-simulated honeypot deployment engine. This honeyd honeypot can respond to a remote attacker who tries to contact the SMTP service with fake responses.



```
Echo "220 intranet ESMTP
sendmail 8.1"

While read data
{
    if data ~ "HELO" then…
    if data ~ "MAIL FROM" then…
…
}
```

Figure 12.57: Honeyd fake response

An attacker can identify the presence of honeyd honeypot by performing time-based TCP fingerprinting methods (SYN proxy behavior). The following figure shows the difference between a response to a normal computer and the response of honeyd honeypot to a manual SYN request sent by an attacker.

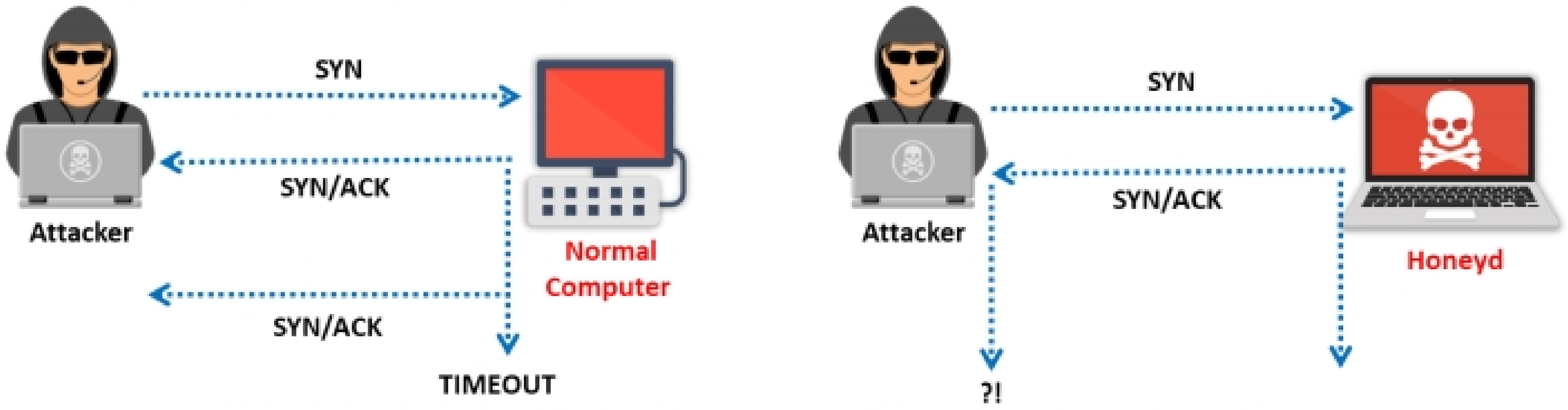


Figure 12.58: Response to SYN request by normal computer vs. Honeyd Honeypot

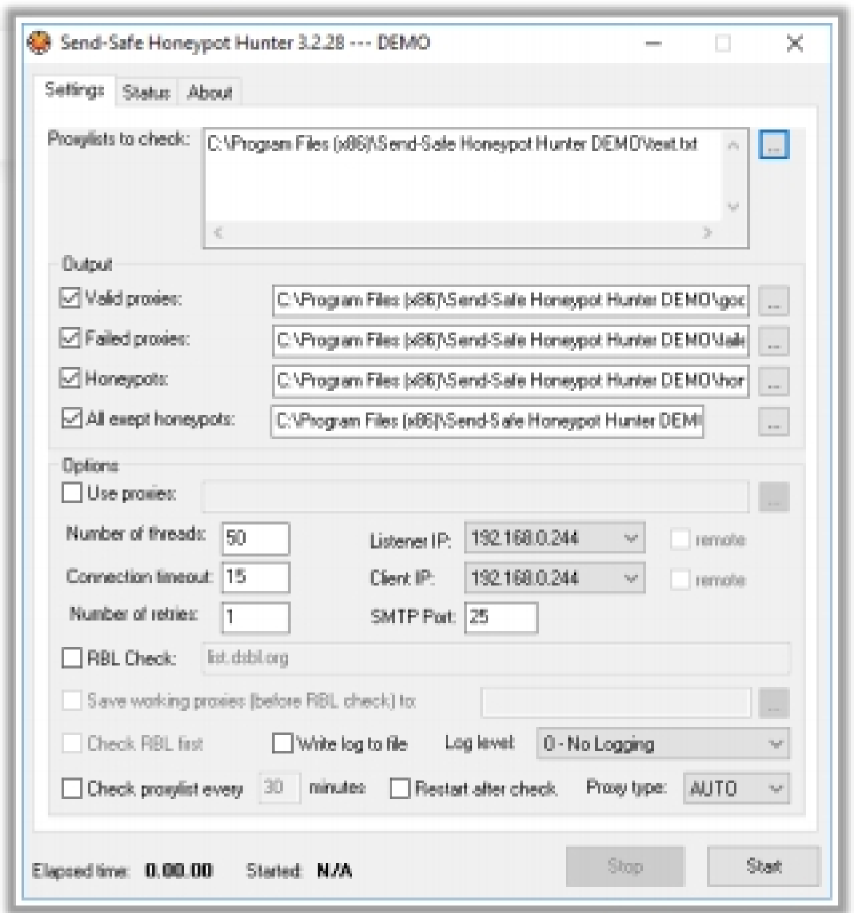
- **Detecting the presence of User-Mode Linux (UML) Honeypot**: User-Mode Linux is an open-source software under GNU, which is used to create virtual machines and is efficient in deploying honeypots. Attackers can identify the presence of UML honeypots by analyzing files such as /proc/mounts, /proc/interrupts, and /proc/cmdline, which contain UML-specific information.
- **Detecting the presence of Sebek-based Honeypots**: Sebek is a server/client-based honeypot application that captures the rootkits and other malicious malware that hijacks the read() system call. Such honeypots record all the data accessed via reading () call. Attackers can detect the existence of Sebek-based honeypots by analyzing the congestion in the network layer, as Sebek data communication is usually unencrypted. Since Sebek logs everything that is accessed via reading () call before transferring to the network, it causes the congestion effect.
- **Detecting the presence of Snort_inline Honeypot**: Snort_inline is a modified version of Snort IDS that is capable of packet manipulation. It can rewrite rules in the iptables and is mainly used in GenII (2nd generation) honeynets to block known attacks and avoid attacker bouncing. Attackers can identify these honeypots by analyzing the outgoing packets. If an outgoing packet is dropped, it might look like a black hole to an attacker, and when the snort_inline modifies an outgoing packet, the attacker can capture the modified packet through another host system and identify the packet modification.
- **Detecting the presence of Fake AP**: Fake access points are those that create fake 802.11b beacon frames with randomly generated ESSID and BSSID (MAC address) assignments. Fake access points only send beacon frames but do not produce any fake traffic on the access points, and an attacker can monitor the network traffic and quickly note the presence of fake AP.
- **Detecting the presence of Bait and Switch Honeypots**: Bait and switch honeypots actively participate in security mechanisms that are employed to respond quickly to incoming threats and malicious attempts. They redirect all malicious network traffic to a honeypot after any intrusion attempt is detected. An attacker can identify the presence of such honeypots by looking at specific TCP/IP parameters such as the Round-Trip Time (RTT), the Time To Live (TTL), and the TCP timestamp.

Honeypot Detection Tools: Send-Safe Honeypot Hunter

- Send-Safe Honeypot Hunter is a tool designed for checking **lists of HTTPS and SOCKS proxies** for "honey pots"

Features:

1. Checks lists of **HTTPS**, **SOCKS4**, and **SOCKS5 proxies** with any ports
2. Checks **several remote or local proxylists** at once
3. Can upload **"Valid proxies"** and **"All except honeypots"** files to FTP
4. Can process **proxylists** automatically every specified time interval
5. May be used for **usual proxylist** validating as well



Proxylists to check: C:\Program Files (x86)\Send-Safe Honeypot Hunter DEMO\test.txt
☑ Valid proxies: C:\Program Files (x86)\Send-Safe Honeypot Hunter DEMO\goo
☑ Failed proxies: C:\Program Files (x86)\Send-Safe Honeypot Hunter DEMO\fail
☑ Honeypots: C:\Program Files (x86)\Send-Safe Honeypot Hunter DEMO\hon
☑ All exept honeypots: C:\Program Files (x86)\Send-Safe Honeypot Hunter DEM
☐ Use proxies:
Number of threads: 50
Listener IP: 192.168.0.244 ☐ remote
Connection timeout: 15
Client IP: 192.168.0.244 ☐ remote
Number of retries: 1
SMTP Port: 25
☐ RBL Check: list.dsbl.org
☐ Save working proxies (before RBL check) to:
☐ Check RBL first
☐ Write log to file
Log level: 0 - No Logging
☐ Check proxylist every 30 minutes
☐ Restart after check
Proxy type: AUTO
Elapsed time: 0.00.00
Started: N/A

http://www.send-safe.com

Honeypot Detection Tools

Attackers user honeypot detection tools such as Send-Safe Honeypot Hunter (*http://www.send-safe.com*) and kippo_detect (*https://github.com*) to detect honeypots in the target organizational networks.

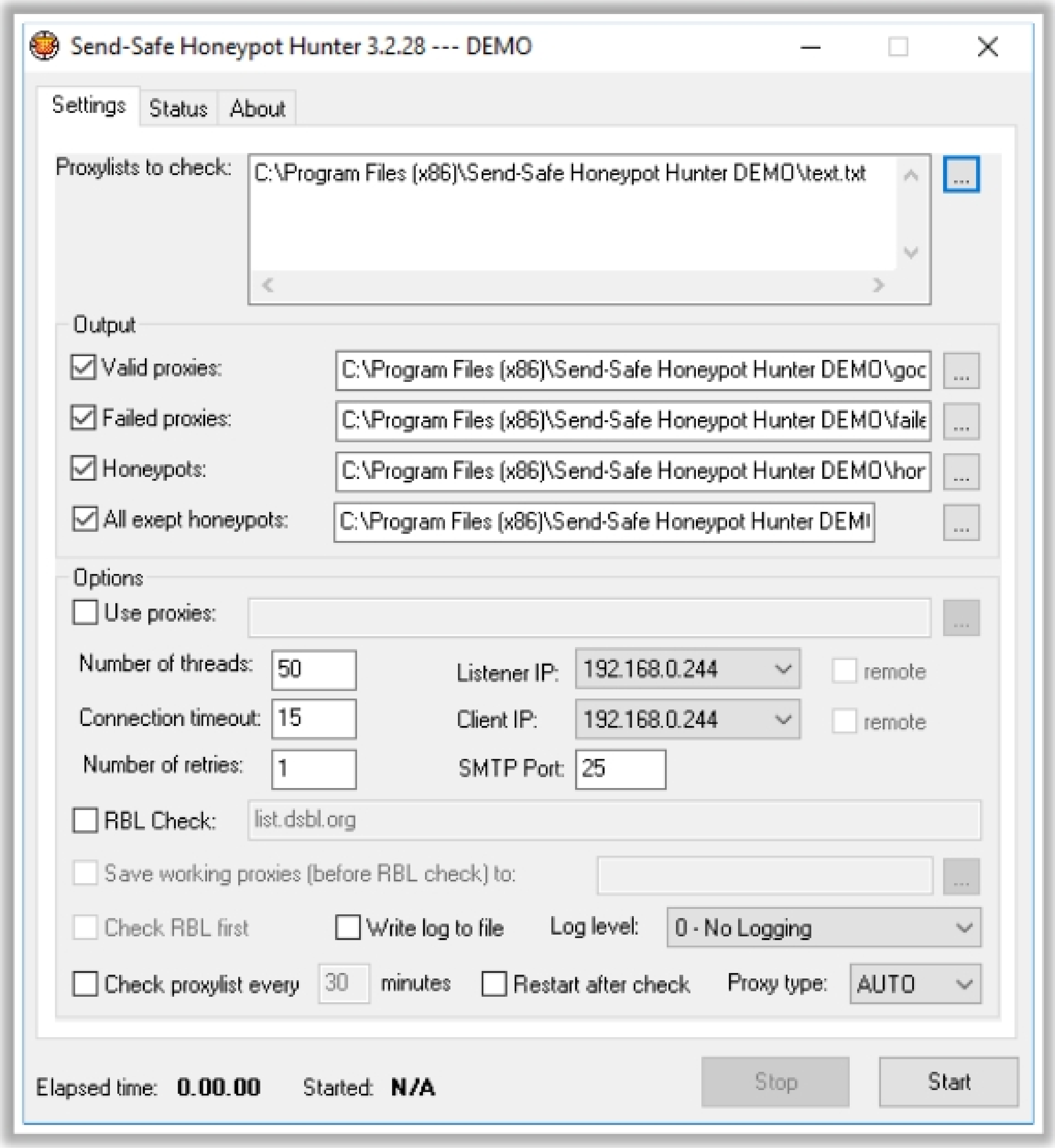
- **Send-Safe Honeypot Hunter**

  Source: *http://www.send-safe.com*

  Send-Safe Honeypot Hunter is a tool designed for checking lists of HTTPS and SOCKS proxies for "honey pots."

  Features:

  - Checks lists of HTTPS, SOCKS4, and SOCKS5 proxies with any ports
  - Checks several remote or local proxylists at once
  - Can upload "Valid proxies" and "All except honeypots" files to FTP
  - Can process proxylists automatically in every specified period
  - May be used for usual proxylist validating as well



Proxylists to check: C:\Program Files (x86)\Send-Safe Honeypot Hunter DEMO\text.txt

☑ Valid proxies: C:\Program Files (x86)\Send-Safe Honeypot Hunter DEMO\goc

☑ Failed proxies: C:\Program Files (x86)\Send-Safe Honeypot Hunter DEMO\faile

☑ Honeypots: C:\Program Files (x86)\Send-Safe Honeypot Hunter DEMO\hor

☑ All exept honeypots: C:\Program Files (x86)\Send-Safe Honeypot Hunter DEMI

☐ Use proxies:

Number of threads: 50

Connection timeout: 15

Number of retries: 1

Listener IP: 192.168.0.244 ☐ remote

Client IP: 192.168.0.244 ☐ remote

SMTP Port: 25

☐ RBL Check: list.dsbl.org

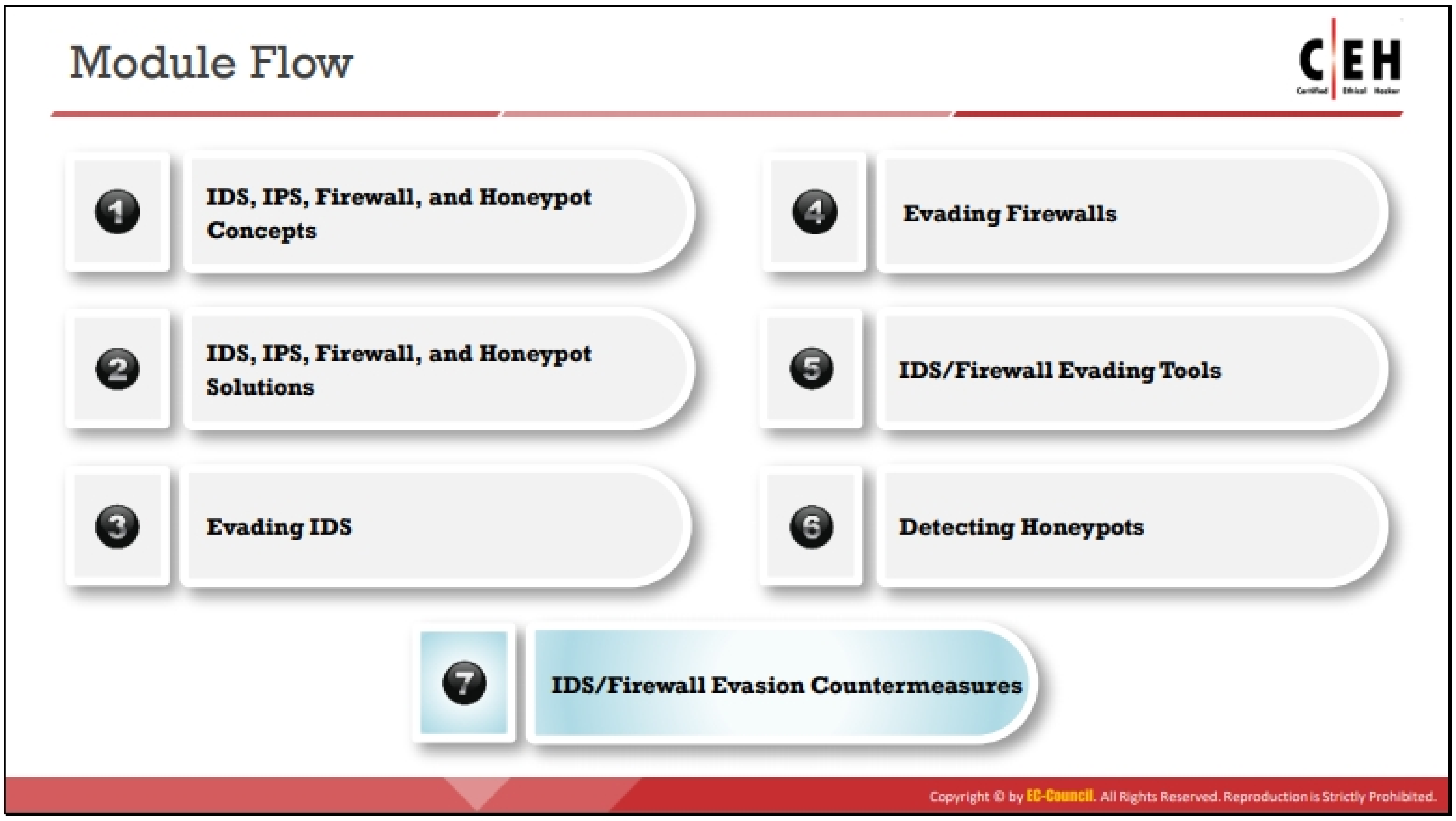
☐ Save working proxies (before RBL check) to:

☐ Check RBL first ☐ Write log to file Log level: 0 - No Logging

☐ Check proxylist every 30 minutes ☐ Restart after check Proxy type: AUTO

Elapsed time: **0.00.00** Started: **N/A**

Figure 12.59: Screenshot of Send-Safe Honeypot Hunter

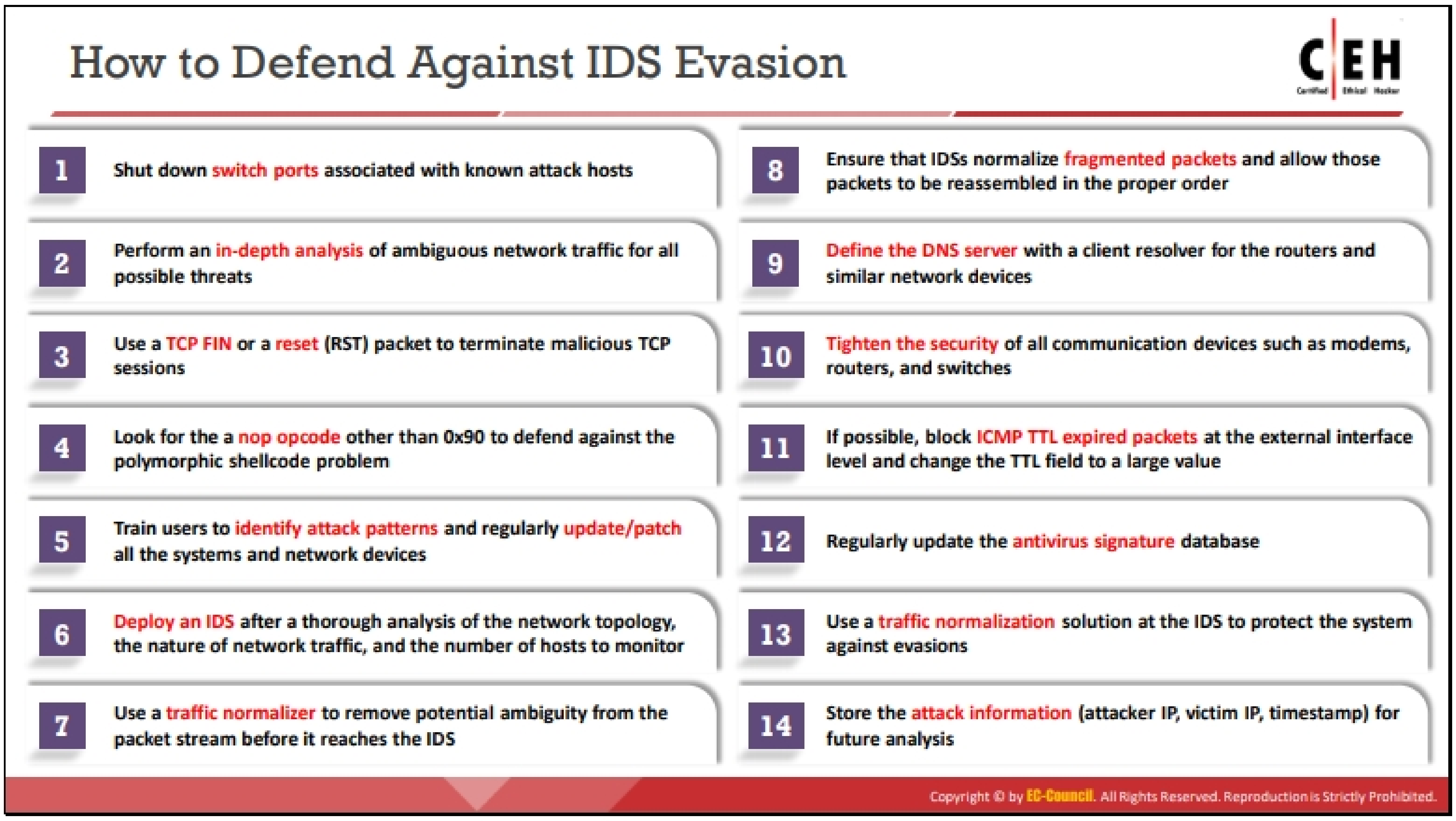


Module Flow

1. **IDS, IPS, Firewall, and Honeypot Concepts**
2. **IDS, IPS, Firewall, and Honeypot Solutions**
3. **Evading IDS**
4. **Evading Firewalls**
5. **IDS/Firewall Evading Tools**
6. **Detecting Honeypots**
7. **IDS/Firewall Evasion Countermeasures**

IDS/Firewall Evasion Countermeasures

The previous sections discussed various tools and techniques used by attackers to bypass network security perimeters such as IDS, firewalls, and honeypots to enter target networks. It is necessary to deploy and configure these security mechanisms securely to avoid attacks. This section discusses various countermeasures and best practices for hardening such network security perimeters.

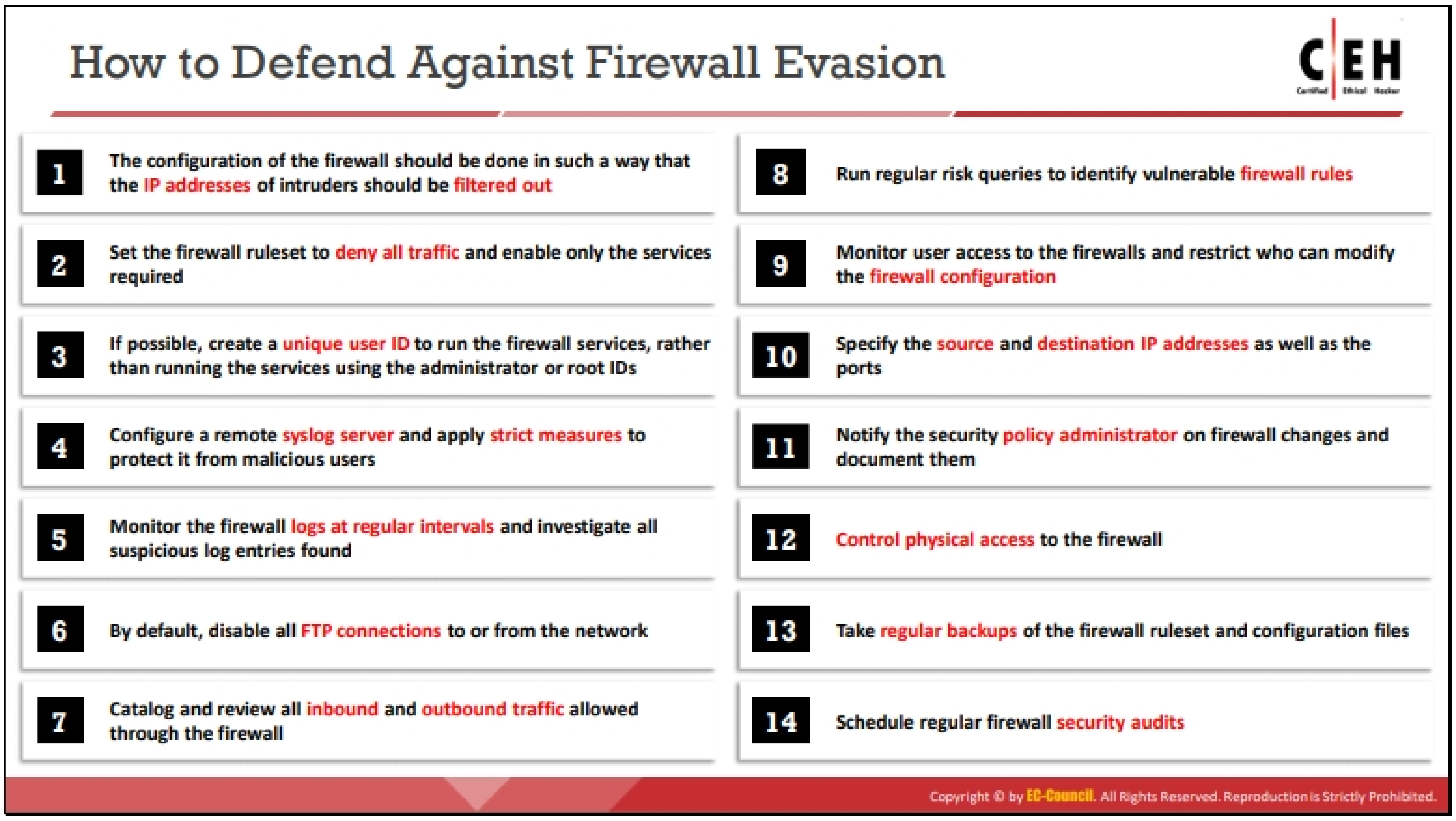


How to Defend Against IDS Evasion

1. Shut down switch ports associated with known attack hosts
2. Perform an in-depth analysis of ambiguous network traffic for all possible threats
3. Use a TCP FIN or a reset (RST) packet to terminate malicious TCP sessions
4. Look for the a nop opcode other than 0x90 to defend against the polymorphic shellcode problem
5. Train users to identify attack patterns and regularly update/patch all the systems and network devices
6. Deploy an IDS after a thorough analysis of the network topology, the nature of network traffic, and the number of hosts to monitor
7. Use a traffic normalizer to remove potential ambiguity from the packet stream before it reaches the IDS
8. Ensure that IDSs normalize fragmented packets and allow those packets to be reassembled in the proper order
9. Define the DNS server with a client resolver for the routers and similar network devices
10. Tighten the security of all communication devices such as modems, routers, and switches
11. If possible, block ICMP TTL expired packets at the external interface level and change the TTL field to a large value
12. Regularly update the antivirus signature database
13. Use a traffic normalization solution at the IDS to protect the system against evasions
14. Store the attack information (attacker IP, victim IP, timestamp) for future analysis

How to Defend Against IDS Evasion

- Shut down switch ports associated with known attack hosts.
- Perform an in-depth analysis of ambiguous network traffic for all possible threats.
- Use TCP FIN or Reset (RST) packet to terminate malicious TCP sessions.
- Look for the nop opcode other than 0x90 to defend against the polymorphic shellcode problem.
- Train users to identify attack patterns and regularly update/patch all the systems and network devices.
- Deploy IDS after a thorough analysis of the network topology, nature of network traffic, and number of hosts to monitor.
- Use a traffic normalizer to remove potential ambiguity from the packet stream before it reaches the IDS.
- Ensure that IDS normalize fragmented packets and allow those packets to be reassembled in the proper order.
- Define DNS server for client resolver in routers or similar network devices.
- Harden the security of all communication devices such as modems, routers, etc.
- If possible, block ICMP TTL expired packets at the external interface level and change the TTL field to a considerable value, ensuring that the end host always receives the packets.
- Regularly update the antivirus signature database.
- Use a traffic normalization solution at the IDS to protect the system from evasions.
- Store the attack information (attacker IP, victim IP, timestamp) for future analysis.

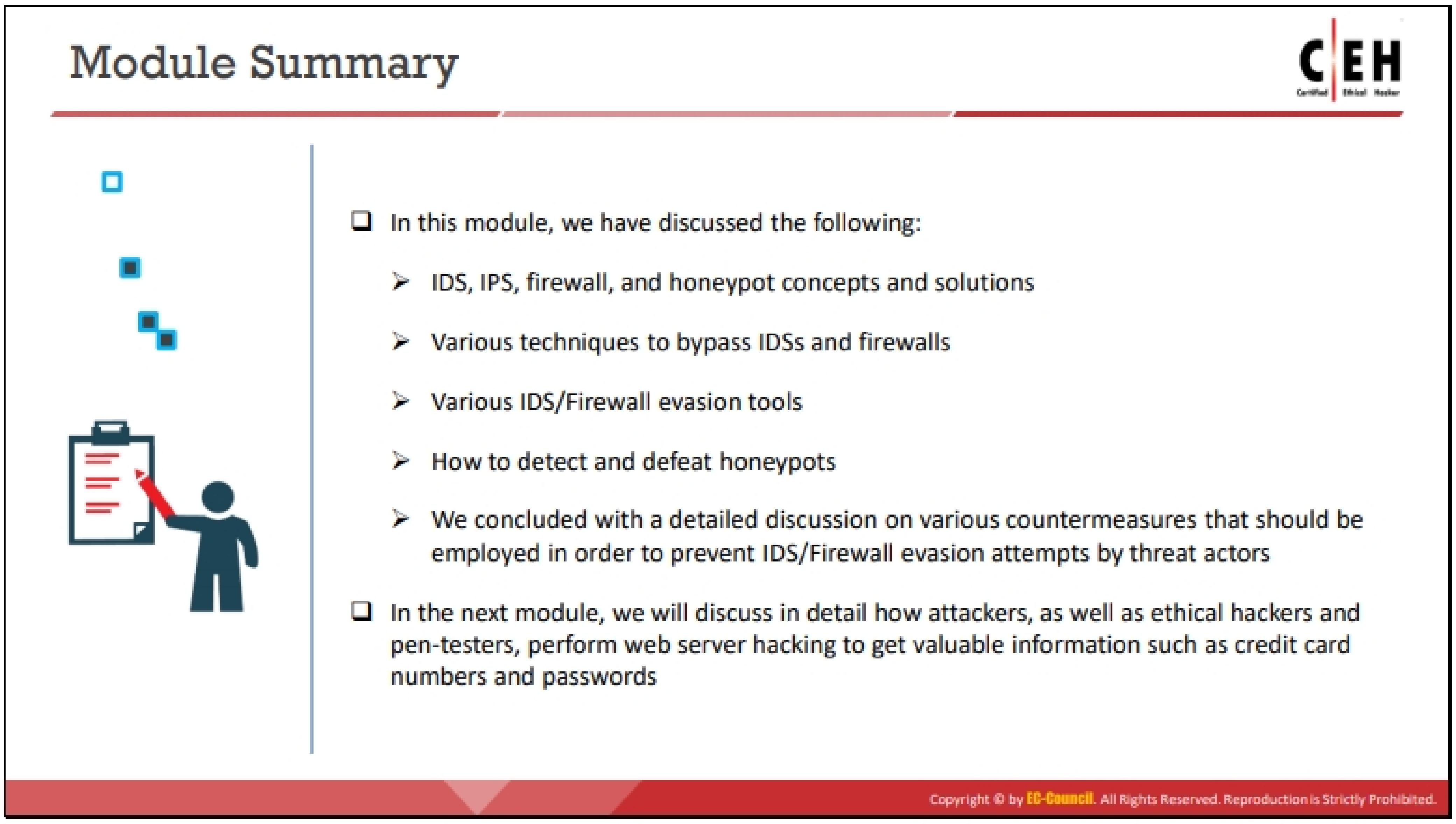


How to Defend Against Firewall Evasion

1. The configuration of the firewall should be done in such a way that the IP addresses of intruders should be filtered out
2. Set the firewall ruleset to deny all traffic and enable only the services required
3. If possible, create a unique user ID to run the firewall services, rather than running the services using the administrator or root IDs
4. Configure a remote syslog server and apply strict measures to protect it from malicious users
5. Monitor the firewall logs at regular intervals and investigate all suspicious log entries found
6. By default, disable all FTP connections to or from the network
7. Catalog and review all inbound and outbound traffic allowed through the firewall
8. Run regular risk queries to identify vulnerable firewall rules
9. Monitor user access to the firewalls and restrict who can modify the firewall configuration
10. Specify the source and destination IP addresses as well as the ports
11. Notify the security policy administrator on firewall changes and document them
12. Control physical access to the firewall
13. Take regular backups of the firewall ruleset and configuration files
14. Schedule regular firewall security audits

How to Defend Against Firewall Evasion

- The firewall should be configured such that the IP address of an intruder should be filtered out.
- Set the firewall rule set to deny all traffic and enable only the services required.
- If possible, create a unique user ID to run the firewall services instead of running the services using the administrator or root ID.
- Configure a remote syslog server and adopt strict measures to protect it from malicious users.
- Monitor firewall logs at regular intervals and investigate all suspicious log entries found.
- By default, disable all FTP connections to or from the network.
- Catalog and review all inbound and outbound traffic allowed through the firewall.
- Run regular risk queries to identify vulnerable firewall rules.
- Monitor user access to firewalls and control who can modify the firewall configuration.
- Specify the source and destination IP addresses as well as the ports.
- Notify the security policy administrator about firewall changes and document them.
- Control physical access to the firewall.
- Take regular backups of the firewall rule set and configuration files.
- Schedule regular firewall security audits.



Module Summary

- In this module, we have discussed the following:
  - IDS, IPS, firewall, and honeypot concepts and solutions
  - Various techniques to bypass IDSs and firewalls
  - Various IDS/Firewall evasion tools
  - How to detect and defeat honeypots
  - We concluded with a detailed discussion on various countermeasures that should be employed in order to prevent IDS/Firewall evasion attempts by threat actors
- In the next module, we will discuss in detail how attackers, as well as ethical hackers and pen-testers, perform web server hacking to get valuable information such as credit card numbers and passwords

Module Summary

This module discussed different IDS, IPS, firewall, and honeypot concepts and solutions. It also described various techniques for bypassing IDS and firewalls. In addition, it illustrated various IDS/firewall evasion tools. Further, it explained how to detect and defeat honeypots. Finally, it ended with a detailed discussion of various countermeasures to be adopted to prevent IDS/Firewall evasion attempts by threat actors.

In the next module, we will discuss in detail how attackers as well as ethical hackers and pen-testers perform web server hacking to gain valuable information such as credit card numbers and passwords.